UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN VISUAL PARA LA CREACIÓN DE ENTORNOS DE SIMULACIÓN DISTRIBUIDOS

Autor:

Miguel Ángel Hernández Acero

Directores:

Alberto Núñez Covarrubias

Laura Prada Camacho

Fecha de terminación: Julio 2012

Agradecimientos

En esta travesía que se ha hecho especialmente ardua en la recta final, he de agradecer el apoyo y la compresión a mis padres y a mi hermano. Sin su paciencia y ánimo no hubiera sido lo mismo. Gracias por todo.

A mi tutor Alberto, por estar siempre cuando lo he necesitado. Por mostrarme a abordar cada problema a su tiempo y a enfrentarlo de la manera más sencilla. A mi tutora Laura, que se unió en la parte final, por sus consejos, por sus ánimos y por todas las gestiones.

A todos los amigos que he hecho por el camino: a Javier, a Alberto, a Álvaro M., a Adolfo, a Fernando, a Álvaro V., a Francisco Javier, a Sergio, a Alex y a Carlos. Por todos esos momentos dentro y fuera de la universidad, absolutamente incatalogables.

Por supuesto a Fer y a Miguel, por ese día a día, por esas situaciones épicas, por esas risas y por todos aquellos buenos momentos que me llevo.

Y al resto de compañeros, que de una forma u otra han compartido esta aventura.

Muchas gracias a todos.

Índice General

1 INTRODUCCIÓN 11

1.1 OBJETIVOS 12

1.2 ESTRUCTURA DEL DOCUMENTO 13

2 ESTADO DEL ARTE 15

2.1 OMNET++ 15

2.2 PARSEC 17

2.3 DESMO-J 18

2.4 JAVASIM 18

2.5 ADEVS 19

2.6 OPNET 19

2.7 XML 20

3 PLATAFORMA DE SIMULACIÓN SIMCAN 31

3.1 INTRODUCCIÓN 32

3.2 ARQUITECTURA DEL SISTEMA 34

3.3 CARACTERÍSTICAS 38

4 ANÁLISIS 41

4.1 INTRODUCCIÓN 41

4.2 DESCRIPCIÓN DE LA APLICACIÓN 42

4.3 ELEMENTOS DEL ESCENARIO 43

4.3.1 Nodos 43

4.3.2 Switches 48

4.3.3 Racks 48

4.4 FUNCIONALIDADES A DESARROLLAR 49

4.5 DESCRIPCIÓN DE REQUISITOS 51

4.6 CASOS DE USO 69

5 DISEÑO 77

5.1 INTRODUCCIÓN 77

5.2 MODELO LÓGICO 78

5.3 FLUJO DE INFORMACIÓN DEL SISTEMA 80

5.4 ANALIZADOR SINTÁCTICO 81

5.5 INTERFAZ DE USUARIO 84

5.6 SINTAXIS DEL FICHERO DE CONFIGURACIÓN DEL SISTEMA 101

5.7 SINTAXIS DE LOS FICHEROS DE ALMACENAMIENTO DE LOS ELEMENTOS DEL SISTEMA 106

5.8 SINTAXIS DE LOS FICHEROS DE SALIDA 111

5.8.1 Omnet.ini 112

5.8.2 Scenario.ned 114

6 CONCLUSIONES Y TRABAJO FUTURO 119

ANEXO I. PRESUPUESTO 121

ANEXO II. CASO PRÁCTICO 127

FICHEROS GENERADOS 130

ANEXO III BIBLIOGRAFÍA 145

Índice de Ilustraciones

Ilustración 1 Capas del framework 31
Ilustración 2 Esquema de SIMCAN 34
Ilustración 3 Paquetes SIMCAN 37
Ilustración 4: Arquitectura global de SIMCAN 37
Ilustración 5 Casos de uso principales 70
Ilustración 6 Casos de uso derivados de la gestión de escenarios 71
Ilustración 7 Casos de usos derivados de la gestión de los nodos 72
Ilustración 8 Casos de usos derivados de la gestión delos racks 72
Ilustración 9 Casos de usos derivados de la gestión de los switches 73
Ilustración 10 Casos de usos derivados de la gestión del repositorio SIMCAN 73
Ilustración 11 Casos de uso derivados de la gestión de los elementos del escenario 74
Ilustración 12 Casos de uso derivados de la gestión de la simulación 75
Ilustración 13 Modelo lógico del sistema 79
Ilustración 14 Flujo de información del sistema 81
Ilustración 15 Modelo del scanner 83
Ilustración 16 Pop up menú del repositorio 86
Ilustración 17 Pantalla principal 86
Ilustración 18 Pop up menú del escenario 88
Ilustración 19 Renombrar elemento 88
Ilustración 20 Cambiar imagen del elemento 89
Ilustración 21 Configurador de conexión 90
Ilustración 22 Eliminar una conexión 91
Ilustración 23 Escaneo de directorios 92
Ilustración 24 Configuración del nodo 93
Ilustración 25 Configuración de Sistema Operativo. 94
Ilustración 26 Configuración del sistema de archivos 95
Ilustración 27 Administrador de volúmenes 96
Ilustración 28 Configuración del módulo CPU 98
Ilustración 29 Configuración del módulo Block Server 98
Ilustración 30 Configuración del módulo de Aplicaciones 99
Ilustración 31 Configuración de un Switch 100
Ilustración 32 Configuración de un Rack 101
Ilustración 33 Fichero CPU_Module.ned 102
Ilustración 34 Fichero de almacenamiento de un Nodo 107
Ilustración 35 Detalle de aplicaciones de un nodo 108
Ilustración 36 Detalle de block server de un nodo 109
Ilustración 37 Detalle de CPU de un nodo 110

Ilustración 38 Diagrama de Gantt 122
Ilustración 39 Pantalla de generación del fichero del sistema 127
Ilustración 40 Pantalla crear nuevo escenario 128
Ilustración 41 Pantalla de escenario caso práctico 129
Ilustración 42 Fichero del sistema 130
Ilustración 43 Detalle del SO de un nodo 131
Ilustración 44 Cabecera del fichero de salida omnet.ini 132
Ilustración 45 Definición de un elemento nodo en el fichero omnet.ini 134
Ilustración 46 Definición de un elemento rack en el fichero omnet.ini 135
Ilustración 47 Definición de un elemento switch en el fichero omnet.ini 136
Ilustración 48 Ranks de los racks en el fichero omnet.ini 137
Ilustración 49 Asignación de direcciones MAC en el fichero omnet.ini 140
Ilustración 50 Cabecera del fichero scenario.ned 141
Ilustración 51 Configuración de red en el fichero scenario.ned 142
Ilustración 52 Conexiones establecidas recogidas en el fichero scenario.ned 144

Índice de Tablas

TABLA 1 COMPARATIVA ENTRE FRAMEWORKS DE SIMULACIÓN 20
TABLA 2 FUNCIONALIDAD GESTIONAR ESCENARIOS 52
TABLA 3 FUNCIONALIDAD GESTIONAR REPOSITORIO SIMCAN 52
TABLA 4 FUNCIONALIDAD GESTIONAR NODOS 52
TABLA 5 FUNCIONALIDAD GESTIONAR SWITCHES 53
TABLA 6 FUNCIONALIDAD GESTIONAR RACKS 53
TABLA 7 FUNCIONALIDAD GESTIONAR ELEMENTOS DEL ESCENARIO 53
TABLA 8 FUNCIONALIDAD GESTIONAR SIMULACIÓN 54
TABLA 9 FUNCIONALIDAD AÑADIR CONEXIÓN 54
TABLA 10 FUNCIONALIDAD ELIMINAR CONEXIÓN 55
TABLA 11 FUNCIONALIDAD RENOMBRAR ELEMENTO 55
TABLA 12 FUNCIONALIDAD CAMBIAR ICONO 56
TABLA 13 FUNCIONALIDAD ELIMINAR ELEMENTO DEL ESCENARIO 56
TABLA 14 FUNCIONALIDAD ESCANEAR DIRECTORIO 57
TABLA 15 GUARDAR FICHERO DE REPOSITORIO 57
TABLA 16 FUNCIONALIDAD CARGAR FICHERO DE REPOSITORIO 58
TABLA 17 FUNCIONALIDAD CREAR NODO 58
TABLA 18 FUNCIONALIDAD EDITAR NODO 59
TABLA 19 FUNCIONALIDAD ELIMINAR NODO 59
TABLA 20 FUNCIONALIDAD AÑADIR NODO AL ESCENARIO 60
TABLA 21 FUNCIONALIDAD AÑADIR N NODOS AL ESCENARIO 60
TABLA 22 FUNCIONALIDAD AÑADIR SWITCH 61
TABLA 23 FUNCIONALIDAD EDITAR SWTICH 61
TABLA 24 FUNCIONALIDAD ELIMINAR SWITCH 62
TABLA 25 FUNCIONALIDAD AÑADIR SWITCH AL ESCENARIO 62
TABLA 26 FUNCIONALIDAD AÑADIR N SWITCHES AL ESCENARIO 63
TABLA 27 FUNCIONALIDAD CREAR RACK 63
TABLA 28 FUNCIONALIDAD EDITAR RACK 64
TABLA 29 FUNCIONALIDAD ELIMINAR RACK 64
TABLA 30 FUNCIONALIDAD AÑADIR RACK AL ESCENARIO 65
TABLA 31 FUNCIONALIDAD AÑADIR N RACKS AL ESCENARIO 65
TABLA 32 FUNCIONALIDAD NUEVO ESCENARIO 66
TABLA 33 FUNCIONALIDAD GUARDAR ESCENARIO 66
TABLA 34 FUNCIONALIDAD GUARDAR ESCENARIO COMO 67
TABLA 35 FUNCIONALIDAD ESCENARIO 67
TABLA 36 FUNCIONALIDAD GENERAR ARCHIVOS DE CONFIGURACIÓN 68
TABLA 37 FUNCIONALIDAD HABILITAR SIMULACIÓN PARALELA 68

TABLA 38 FUNCIONALIDAD LANZAR SIMULACIÓN 69
TABLA 39 DESGLOSE POR ACTIVIDADES DEL PROYECTO 121
TABLA 40 SALARIOS POR CATEGORÍA 123
TABLA 41 COSTE TOTAL PERSONAL 124
TABLA 42 RECURSOS MATERIALES EMPLEADOS 125
TABLA 43 RESUMEN DEL PRESUPUESTO 126

1 INTRODUCCIÓN

La creciente complejidad de los sistemas de computación ha hecho que los simuladores sean una elección importante para el diseño y análisis de arquitecturas complejas a gran escala. Debido al amplio número de campos en la arquitectura de computadores, el desarrollo de un simulador universal es poco práctico e inviable. Naturalmente los investigadores quieren simular el sistema completo con total precisión, pero existen serias dificultades al respecto: alto coste, tiempo para completarlo, imprecisiones en la especificación y errores de implementación.

La creación de entornos de simulación distribuidos requiere la configuración de un número elevado de parámetros, el cual aumenta de manera proporcional al tamaño del entorno a simular.

En la creación y configuración de entornos distribuidos de gran escala surgen varios inconvenientes. Uno de ellos consiste en crear los ficheros de configuración, donde en la mayoría de los casos se realizan de forma manual, debido a que son muy costosos de configurar tanto en tiempo como en esfuerzo. A esto hay que añadir el tiempo dedicado a la fase de detección y corrección de errores. Al realizar la edición de los ficheros de configuración de manera manual, es fácil perder la coherencia global debido a que existen parámetros que dependen de otros. También se pueden producir erratas que hay que localizar y corregir. Para entornos pequeños es sencillo detectar y corregir los errores, pero si el entorno está constituido por miles de nodos, tanto la elaboración como la detección de errores de los ficheros de configuración se convierten en tareas complejas y tediosas.

Otro aspecto que dificulta la creación de este tipo de entornos es conocer todos los módulos existentes a la hora de realizar una simulación. Existen múltiples

módulos para cada componente del sistema a modular como unidades de disco, sistemas de archivos, administradores de volúmenes, planificadores, memorias caché, redes de comunicación, dispositivos de comunicación, etc. En un sistema distribuido a gran escala el número de combinaciones posibles es muy elevado, por esta razón realizar la configuración sin ningún tipo de soporte visual lo hace aún más complejo.

Este proyecto consiste en el diseño e implementación de una aplicación que facilite la gestión de la configuración de entornos de simulación distribuidos a gran escala. La aplicación se centra en una herramienta visual que permita al usuario modelar entornos así como parametrizar los distintos módulos que lo forman.

1.1 Objetivos

El objetivo fundamental de este proyecto es diseñar e implementar una herramienta que facilite la configuración y generación de entornos distribuidos de simulación. Los objetivos que se pretenden alcanzar son:

- La gestión de forma jerárquica y estructurada de todos los módulos de un simulador. Además, exportar e importar estos datos dotaría de flexibilidad a la aplicación. La misma configuración se podría usar en diferentes entornos y permitiría compartir dicha información entre usuarios.

- Permitir al usuario desarrollar o editar el entorno de simulación de manera visual. Así como gestionar cada elemento básico que compone el entorno para construir modelos con un alto nivel de detalle y precisión. Por otra parte, dar la posibilidad al usuario de gestionar los módulos que

han sido configurados. De esta manera permitiría reutilizarlos tantas veces como fuera necesario sin la necesidad de volver a configurar cada vez todos los elementos parametrizables. El usuario no sólo ahorraría tiempo, sino que evitaría cometer errores al utilizar módulos que ya ha probado con anterioridad.

- Visualizar las conexiones que existen entre los componentes, así como permitir su creación o edición de manera intuitiva.

- La aplicación ha de generar los ficheros de configuración correspondientes al entorno distribuido creado. Estos ficheros deben ser creados con la sintaxis *OMNeT++*.

1.2 Estructura del documento

Este documento se ha dividido en 6 capítulos principales, debido a que se han considerado los oportunos para enfocar el proyecto en cada contexto y detallarlo en toda su amplitud.

En este primer capítulo se ha resumido brevemente, el marco en el que se encuadra este proyecto y se han descrito cuales son los objetivos que se desean alcanzar mediante el desarrollo del mismo. En el segundo capítulo, se busca acercar al lector a las posibilidades de los simuladores de sistemas. Se hace un recorrido sobre los principales frameworks de simulación donde se exponen sus características y su ámbito de trabajo. En el tercer capítulo se describe la plataforma de simulación SIMCAN. Se detalla cómo está construida y las funcionalidades que ofrece para los entornos de simulación a gran escala. En el cuarto capítulo se realiza el análisis en profundidad del sistema a desarrollar. Para ello se describe la funcionalidad que se desea implementar a partir de los

requisitos de información que tiene el usuario de la aplicación. En el quinto capítulo se realiza el diseño del sistema a partir del análisis obtenido en el punto anterior, describiendo el modelo lógico del sistema con los componentes que lo forman, el flujo de información, la interfaz de usuario, etc. En el sexto capítulo incluirá una serie de conclusiones del proyecto y líneas a seguir en el futuro. El anexo I contiene el presupuesto del proyecto junto con el diagrama de Gantt En el anexo II se muestra un ejemplo práctico de ejecución de la herramienta.

2 ESTADO DEL ARTE

Actualmente, existen distintos tipos de herramientas o *frameworks* para simular una amplia variedad de sistemas, cada uno adaptado a un tipo de problema específico.

La mayoría de estos *frameworks* están diseñados como lenguajes de programación de alto nivel que permiten al usuario simular distintos tipos de sistemas de una manera flexible. A continuación se detallan los *frameworks* libres más importantes.

2.1 OMNeT++

OMNeT++ es un simulador modular de eventos discretos en C++. Está diseñado para modelar el tráfico de redes de telecomunicaciones, sistemas paralelos y sistemas distribuidos. OMNeT++ está diseñado para realizar simulaciones a gran escala, para lo cual utiliza modelos jerárquicos y componentes reutilizables. La estructura básica de un simulador OMNeT++ se compone de un conjunto de módulos que envían y reciben mensajes a través de puertos de entrada y salida. Los módulos envían y reciben mensajes a través de distintas conexiones predefinidas configuradas entre los módulos. De esta forma las conexiones forman una red que simula la arquitectura deseada.

Los módulos se implementan como objetos C++ que se heredan de una clase base. El comportamiento de los módulos puede ser programado a partir de dos modelos de programación diferentes:

- Modelo orientado a eventos. Cada vez que llegue un mensaje, una función del módulo (*handler*) será ejecutada, utilizando el mensaje como un parámetro de la función.

- Modelo basado en corrutinas. El módulo ejecuta la función principal. Los mensajes son procesados cuando la función principal ejecuta una sentencia de recepción.

El modelo basado en corrutina es más intuitivo y fácil para el desarrollo. El lado negativo es la gran cantidad de memoria que precisa para asignar una pila a cada módulo. Por lo tanto este modelo no es útil para simulaciones a gran escala. De esta manera se deduce que la mejor opción para simulaciones a gran escala es el modelo orientado a eventos.

Los módulos tienen una organización jerárquica que facilita la construcción de una arquitectura de computación en paralelo. Existen dos tipos de módulos:

- Módulos simples: son aquellos que no incluyen nada más que el propio módulo.

- Módulos compuestos: son aquellos que incluyen otros módulos como componentes. Las conexiones asociadas a un módulo compuesto puede ser redirigido a / desde un sub-módulo interno

Los mensajes también se implementan como objetos C++ que se heredan de una clase mensaje base. Los mensajes son una colección de datos y funciones. Estas funciones se utilizan para obtener o establecer los datos y realizar operaciones sobre ellos. Existen diferentes maneras de implementar el objeto mensaje:

- Escribiendo una especificación de los datos del mensaje (usando el formato .msg). Esta especificación es pre-compilada para generar el

objeto C++. Este método sólo funciona con mensajes compuestos por valores simples.

- Escribiendo un objeto C++ que herede de otro generado con la especificación .msg. El nuevo objeto puede modificar la especificación .msg a voluntad.

La simulación se configura mediante la especificación .ned, que describe cuantas instancias de cada módulo tienen que ser simuladas, y además se detallan las conexiones entre todos los módulos. Otra ventaja es la parametrización de los módulos mediante valores almacenados en su propia especificación, como el nombre de cada módulo. La especificación .ned se carga al principio de la ejecución, de esta forma la simulación se puede reconfigurar completamente sin la necesidad de recompilar el código.

2.2 PARSEC

PARSEC es un lenguaje de simulación de eventos discretos. Consiste en un compilador de C mejorado con la capacidad de definir y crear entidades de simulación. Además posee constructores para mensajes de comunicación entre entidades. Fundamentalmente PARSEC consiste de tres elementos principales: un lenguaje de simulación paralela llamada Parsec (entorno de simulación paralela para sistemas complejos, en sus siglas en inglés); su interfaz gráfica de usuario (GUI) llamada Pave; y el sistema de ejecución portable que implementa la simulación de algoritmos. Este simulador tiene varios inconvenientes importantes. El principal es la poca flexibilidad del simulador. Además, los modelos muy grandes no se escalan de forma correcta porque está basado en un sistema de corrutina.

2.3 DESMO-J

DESMO-J es un *framework* orientado a objetos, dirigido a desarrolladores de modelos de simulación. El acrónimo en inglés significa simulador de eventos discretos y modelado en Java (Discrete-Event Simulation and Modeling in Java). Se caracteriza por:

- DESMO-J Se basa en el paradigma de simulación de eventos discretos. En los modelos de este tipo, todos los cambios en el sistema ocurren en puntos discretos en el tiempo. Entre los distintos eventos se asume que el estado del sistema permanece constante. La simulación de eventos discretos es especialmente adecuado para los sistemas cuyos cambios de estado se producen repentinamente y de manera irregular.

- DESMO-J está implementado en Java. Usar este *framework* para crear modelos de simulación conlleva a desarrollarlo en Java.

2.4 Javasim

JavaSim es una implementación en Java de la herramienta de simulación original C++ SIM, que consiste en un simulador de eventos discretos basado en procesos, donde cada entidad de simulación puede ser considerada como un proceso independiente. Las entidades de simulación son representadas por objetos proceso, que son realmente objetos de Java que poseen un hilo de ejecución de control independiente asociados con ellos cuando son creados. Estos objetos interactúan unos con otros a través de mensajes y otras primitivas.

En la mayoría de los casos, un programa de simulación necesita modelar los aspectos de un sistema real que se corresponde con las distintas funciones de

distribución. JavaSim proporciona un generador de números aleatorios que siguen cinco funciones comunes de distribución:

1. Distribución distribuida.
2. Distribución exponencial.
3. Distribución Erlang.
4. Distribución Hiper Exponencial.
5. Distribución normal.

Estos generadores aleatorios, junto con los procesos de simulación, constituyen la esencia del paquete JavaSim.

2.5 Adevs

Adevs es una librería de C++ para construir simulaciones de eventos discretos basados en los formalismos Parallel DEVS y Dynamics DEVS. DEVS es un formalismo para modelar y realizar análisis de sistemas de eventos discretos (DESs); fue inventado por Dr.Bernard P.Zeigles. DEVS ha sido utilizado para el estudio de sistemas sociales, sistemas ecológicos, redes de ordenadores y arquitectura de ordenadores, sistemas militares a nivel táctico y en muchas otras áreas.

2.6 OPNET

También existen *frameworks* de simulación comerciales, como por ejemplo OPNET. Este simulador posee una licencia comercial pero distribuye una

licencia especial para estudiantes e investigadores. OPNET proporciona una herramienta para el modelado jerárquico de redes, incluyendo procesos, descripción de la topología de red, y de distintos tipos de simuladores de escenarios de tráfico. Este simulador necesita ser adaptado a entornos síncronos, puesto que requiere un diseño explícito del esquema del reloj y una distribución de red. Uno de los más importantes problemas de este simulador es la falta de realismo de la pila del protocolo de red. Además carece de arquitectura de red ip y de una interfaz para *sockets*.

Framework	Licencia	Apoyo Comunitario	Flexible	Escalable	Paralelo
OMNeT++	APL	Sí	Sí	Sí	Sí
PARSEC	Free non-profit	No	Sí	Sí	Sí
Desmo-J	GPL	No	Sí	No	No
JavaSim	LGPL	No	Sí	No	No
Adevs	Open Source	No	Sí	Sí	Sí
OPNET	Commercial/ Research	Sí	Sí	Sí	Sí

Tabla 1 Comparativa entre frameworks de simulación

2.7 XML

Sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma

manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Ventajas

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

Estructura de un documento XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se

tiene un árbol de pedazos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez con notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de este como un elemento, un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

A continuación se muestra un ejemplo para entender la estructura de un documento XML:

XML Document Example

```
<?xml version="1.0"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Documentos XML bien formados

Se llama documentos "bien formados" (del inglés well formed) a los documentos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, ser analizados correctamente por cualquier "parser" (Analizador Sintáctico) que cumpla con la norma. Se separa esto del concepto de validez que se explica más adelante.

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una

etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.

- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, sólo puede tener un elemento inicial.

- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.

- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos entendibles por las personas.

Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento.

- Prólogo: Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas. El prólogo contiene:

  - una declaración XML. Es la sentencia que declara al documento como un documento XML.
  - una declaración de tipo de documento. Enlaza el documento con su DTD, o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
  - uno o más comentarios e instrucciones de procesamiento.

- Cuerpo: A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener al menos un elemento raíz, característica indispensable también para que el documento esté bien formado.
- Elementos: Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.
- Atributos: Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.
- Entidades predefinidas: Entidades para representar caracteres especiales para que no sean interpretados como marcado en el procesador XML.
- Secciones CDATA: Es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML
- Comentarios: Comentarios a modo informativo para el programador que han de ser ignorados por el procesador.

Validez

Que un documento sea "bien formado" solamente habla de su estructura sintáctica básica, es decir que se componga de elementos, atributos y

comentarios como XML manda que se escriban. Ahora bien, cada aplicación de XML, es decir cada lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.

Esta relación entre elementos se especifica en un documento externo o definición (expresada como DTD (Document Type Definition = Definición de Tipo de Documento) o como XSchema). Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica.

XML es usado como lenguaje base por WSDL y SOAP para llevar a cabo su cometido

XSD

Más conocido como XML Schema Definition. Es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así, una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C) y alcanzó el nivel de recomendación en mayo de 2001.

Terminología

El término "XML Schema" es utilizado con varios significados dentro del mismo contexto de descripción de documentos, y es importante tener en cuenta las siguientes consideraciones:

- "XML Schema" (Esquema XML) es el nombre oficial otorgado a la recomendación del W3C, que elaboró el primer lenguaje de esquema separado de XML (la definición de tipo de documentos (DTD) forma parte de XML).

- Es habitual referirse a los esquemas como "XML schema" de forma genérica, pero se recomienda utilizar el término “documento esquema” (schema document) o "definición de esquema"(schema definition), y reservar “XML Schema” para la denominación de este lenguaje específico.

- Aunque genéricamente se utilice "XML schemas", XSDL (XML Schema Definition Language) es el nombre técnico de los lenguajes de esquema de XML

Componentes

XML Schema es un lenguaje de esquema escrito en XML, basado en la gramática y pensado para proporcionar una mayor potencia expresiva que la DTD, más limitadas en la descripción de los documentos a nivel formal.

Los documentos esquema (usualmente con extensión .xsd de XML Schema Definition (XSD)) se concibieron como una alternativa a las DTD, más compleja, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML. La principal aportación de XML Schema es el gran número de los tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

Los esquemas se construyen a partir de diferentes tipos de componentes:

- Elemento (element)

- Atributo (attribute)

- Tipo simple (simple type)

- Tipo complejo (complex type)
- Notación (notation)
- Grupo modelo nombrado (named model group)
- Grupo de atributos (attribute group)
- Restricción identidad (identity constraint)

Estos componentes ofrecen la posibilidad de combinar características de alto o bajo nivel:

- Alto nivel: Se encargan de ofrecer un significado semántico del contenido del documento. Analizan el contenido y extraen de él un significado. Éste puede estar predefinido en la declaración del esquema o se puede extraer de la misma estructura.
- Bajo nivel: Son características más concretas del documento que están incluidos en los diferentes campos del esquema y se accede a ellas de manera directa. Son los que se comparan directamente con el criterio de búsqueda definido y halla palabras concretas en la definición de los esquemas.

XML Schema supera muchas de las limitaciones y debilidades de las DTDs. Fue diseñado completamente alrededor de namespaces y soporta tipos de datos típicos de los lenguajes de programación, como también tipos personalizados simples y complejos. Un esquema se define pensando en su uso final.

SAX VS DOM

SAX se utiliza para hacer un recorrido secuencial de los elementos del documento XML y DOM implica la creación de un árbol en memoria que contiene el documento XML, y con él en memoria podemos hacer cualquier tipo

de recorrido y acciones con los elementos que queramos. Por lo tanto, se puede observar que la filosofía de trabajo de un parser tipo DOM es totalmente distinta a un parser tipo SAX. Veamos un poco más detalladas ambas filosofías.

DOM (Document Object Model): Los procesadores tipo DOM leen el documento completo y localizan su estructura jerárquica y crean un árbol en memoria RAM que permite acceder a cualquier parte del documento XML en un momento dado. Por lo tanto la forma de construir parsers tipo DOM es muy sencilla ya que para acceder a una parte del documento XML basta con recorrer el árbol generado a partir del propio documento. Sin embargo, la principal desventaja de este modelo es el gran consumo de memoria que supone emplear este tipo de parsers.

SAX (Simple API for XML): Los procesadores tipo SAX procesan el documento o información en XML de una manera muy diferente a DOM. SAX procesa la información por eventos. A diferencia de DOM que genera un árbol jerárquico en memoria, SAX procesa la información en XML conforme esta sea presentada (evento por evento), es decir, manipulando cada elemento a su determinado tiempo conforme va siendo leído, sin incurrir de esta forma en uso excesivo de memoria. SAX es más aconsejable cuanto más crezca el tamaño del documento XML a procesar, ya que el consumo de recursos de usar DOM en estos casos aumenta considerablemente.

La idea general es la siguiente, al principio del programa se lanza un manejador (handler) que será el encargado de disparar los eventos correspondientes. Principalmente existen cuatro tipos de eventos, cuando se inicia una etiqueta, cuando finaliza una etiqueta, cuando leemos el contenido de una etiqueta y cuando se produce el error. Dentro de cada evento, las API´s tipo SAX nos permiten acceder a estructuras de datos donde podemos saber de qué etiqueta se trata y su contenido o el valor de los atributos asociados a ella. De esta forma,

manejando estos cuatro eventos, el programador debe ser capaz de construir un programa que realice las acciones pertinentes a partir del documento XML.

El principal inconveniente de SAX es que no permite modificar o crear ficheros XML y el acceso a los elementos del fichero es secuencial. La aplicación va a utilizar ficheros pequeños, tanto los de configuración de los datos de la plataforma SIMCAN, como los que contengan la información de los distintos elementos que forman el escenario.

Debido a que el sistema requiere la creación, modificación de los ficheros XML, además de lectura dinámica y no secuencial, la mejor opción para el tratamiento XML de esta aplicación es el procesamiento DOM.

3 PLATAFORMA DE SIMULACIÓN SIMCAN

En este capítulo se describirá la plataforma de simulación SIMCAN, que alberga los datos de configuración necesarios para poder explotar la aplicación a desarrollar expuesta en el presente proyecto.

SIMCAN se basa en la construcción de modelos de simulación, mediante la definición y configuración de cada uno de los cuatro sistemas básicos: sistema de procesamiento, sistema de memoria, sistema de almacenamiento y sistema de redes. Se trata de una plataforma de simulación rápida, flexible, escalable y ampliable para modelar y simular sistemas distribuidos y aplicaciones.

La principal ventaja es la flexibilidad y la escalabilidad obtenida para el modelado y la simulación de entornos complejos a gran escala. Estos entornos pueden contener miles de nodos, modelando cada subsistema con el nivel de detalle requerido.

La plataforma de simulación SIMCAN ha sido construida por encima de INET y OMNeT++ *frameworks*. Además, otros simuladores se pueden añadir a la arquitectura del *framework* para incrementar su funcionalidad. En la Ilustración 1 se muestra las capas del *framework* y la interacción entre ellas.

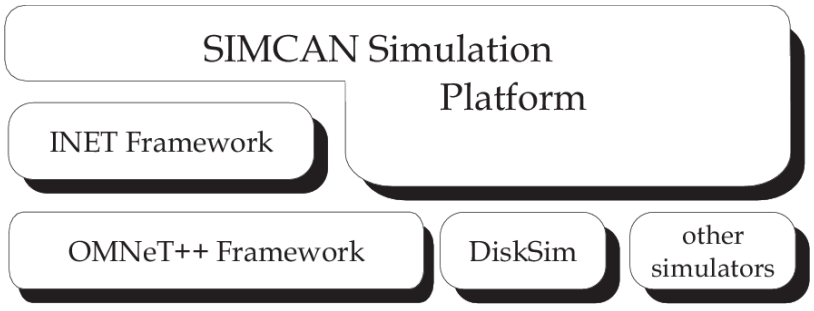

Ilustración 1 Capas del framework

3.1 Introducción

Muchos simuladores existentes son monolíticos o están diseñados para una sola arquitectura. Por esta razón es difícil extraer un componente de la arquitectura del simulador para su reutilización, compartición o comparación. Los investigadores intentan encontrar el simulador que más encaje con su investigación, pero en muchas ocasiones no pueden encontrar ese simulador y se ven abocados a modificar uno existente o codificar uno nuevo.

La filosofía de la estrategia a seguir no es obtener una precisión perfecta, sino permitir un margen mínimo de error en aras de ejecutar las simulaciones mucho más rápido. La precisión absoluta no es siempre estrictamente necesaria y en muchas ocasiones no es ni incluso deseada, debido al alto coste de ingeniería que requiere. En muchas ocasiones, sustituir absoluta precisión por relativa entre distintos simulaciones es suficiente para los usuarios para descubrir tendencias para las técnicas propuestas. Normalmente velocidad y precisión son inversamente proporcionales. Por esta razón, la estrategia propuesta intenta encontrar un ajuste entre ambas características, realizando simulaciones rápidas con un mínimo porcentaje de error.

Las características deseadas para cualquier tipo de estrategia de simulación, son las siguientes:

- Escalabilidad, si la correspondiente estrategia es capaz de simular con suficiente rendimiento sistemas a gran escala, incrementando el número de máquinas, que forman la correspondiente estructura simulada. De esta forma se determina la velocidad a la que el simulador ejecuta la correspondiente simulación. En general, cuanto más grande es el tamaño de la arquitectura a simular, mayor será el tiempo requerido para ejecutar la simulación.

- Una estrategia flexible debe permitir a los usuarios crear un entorno de manera sencilla, mediante el uso de los distintos modelos de componentes con diferentes niveles de detalle. La flexibilidad también indica si un modelo está bien estructurado, puesto que se simplifica la modificación, lo que permite diseñar variantes o incluso realizar diseños completamente diferentes.

- El detalle define el nivel de abstracción usado para implementar los modelos de componentes. Por esta razón los investigadores pueden elegir el nivel de detalle para modelar la arquitectura requerida de forma que el rendimiento sea suficientemente bueno y a la vez suficientemente rápido para modelar sistemas a gran escala y aplicaciones de larga duración en una franja de tiempo aceptable. En general, cuanto más nivel de detalle tenga el simulador, mayor será el retardo.

- Finalmente el término ampliable hace referencia a la capacidad de aumento de la funcionalidad de una plataforma de simulación. En la mayoría de los casos, esta capacidad consiste en añadir nuevos modelos al repositorio de la plataforma de simulación.

SIMCAN permite una estrategia de modelado y simulación de sistemas distribuidos rápida, flexible, escalable y ampliable. Se basa en la integración del modelo de los cuatro sistemas básicos (sistemas de almacenamiento, de procesamiento, de memoria, y de red) dentro de una única plataforma de simulación.

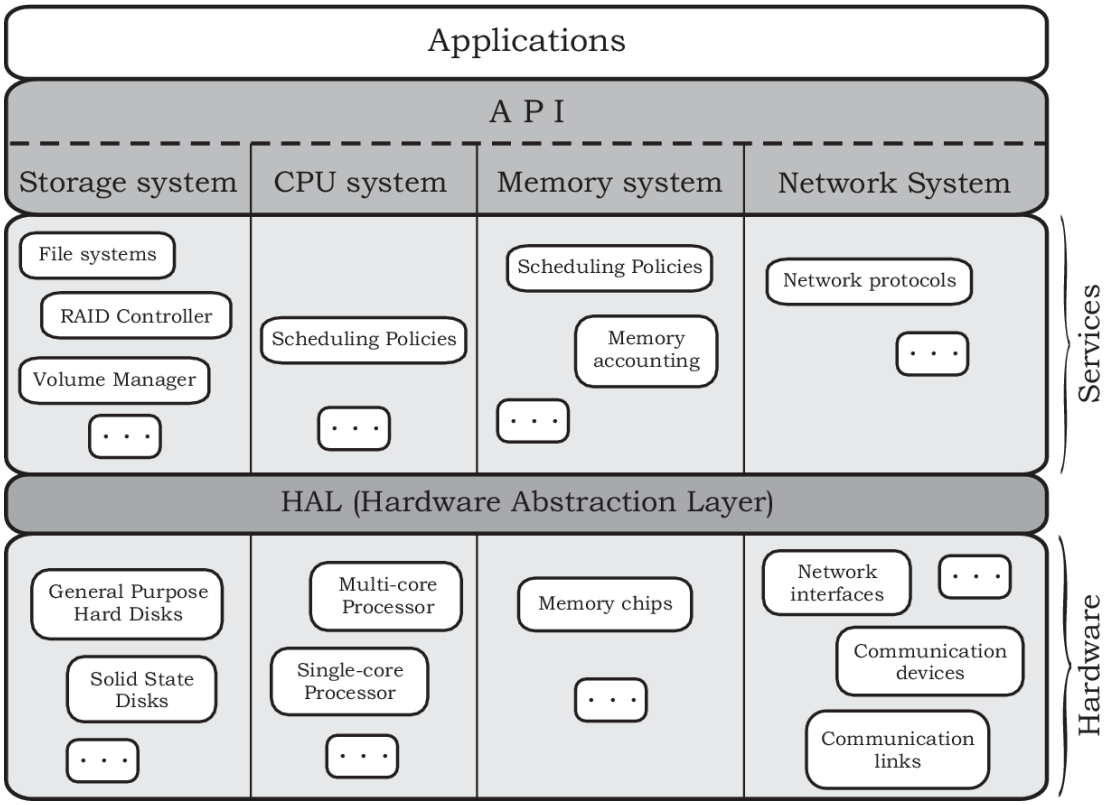

Ilustración 2 Esquema de SIMCAN

3.2 Arquitectura del sistema

SIMCAN ha sido diseñado con el propósito de proveer flexibilidad, precisión, rendimiento y escalabilidad, lo que la convierte en una poderosa plataforma de simulación para diseñar, testear y analizar las arquitecturas. El rango de sistemas a simular va desde un nodo simple hasta a un sistema distribuido completo de gran rendimiento. Lo mejor de esta plataforma de simulación reside en que es capaz de modelar y simular entornos a gran escala (miles de nodos) con un nivel personalizable de detalle.

Las arquitecturas de simulación son modeladas mediante un conjunto de componentes existentes que son provistos por SIMCAM. Estos representan el comportamiento de componentes reales que pertenecen a la arquitectura real

como son los discos, las redes, las memorias, los sistemas de archivos, etc. Estos componentes están organizados de manera jerárquica dentro del repositorio de SIMCAM, que compone el corazón del motor de simulación. Actualmente, SIMCAN provee un amplio rango de componentes para modelar un sistema distribuido completo con diferentes niveles de detalle y escalabilidad.

Además de diseñar entornos de simulación usando componentes provistos por SIMCAN, se pueden añadir nuevos componentes al repositorio. Además, SIMCAN permite de manera sencilla la sustitución de componentes. Estos componentes intercambiables pueden diferir en nivel de detalle (para mejorar el rendimiento en vez de la precisión y viceversa), en el comportamiento funcional o en ambas. Además se pueden incorporar simuladores ya existentes o nuevos al repositorio de SIMCAN, como diskSim. Esto lo convierte en una potente característica para la plataforma de simulación porque una vez que son añadidos al repositorio los nuevos componentes, su funcionalidad se incrementa de la misma manera.

La manera de ejecutar SIMCAN para realizar la correspondiente simulación dependerá de los requisitos del usuario y los recursos disponibles. Además, SIMCAN puede ser ejecutado en un ordenador mediante simulación secuencial o, por el contrario, puede ser ejecutado en paralelo por medio de ordenadores con memoria compartida y sistemas de memoria distribuida. La velocidad de la simulación dependerá en gran medida de los recursos computacionales usados en la ejecución de la simulación. Cuanta más CPU y recursos de memoria estén disponibles, mayor rendimiento se obtendrá en la ejecución de la simulación. Sin embargo, la manera en que se paraleliza la simulación entre las distintas CPUs dependerá de la configuración realizada por el usuario.

Con el propósito de facilitar la tarea de construir y configurar entornos distribuidos a gran escala, la plataforma SIMCAN ofrece una clasificación flexible para la incorporación de bloques de nodos que imita a los utilizados en

sistemas reales. A continuación se describen un conjunto de bloques de nodos provistos por SIMCAN, que permiten al usuario crear las arquitecturas más comunes:

- Nodo de computación: Este módulo simula el comportamiento de un nodo. Contiene los módulos necesarios para simular los sistemas requeridos (ver la Ilustración 2) que se encuentran en un nodo real. Los componentes de cada nodo pueden ser totalmente personalizados y configurados para comportarse como un nodo de computación, como un nodo de almacenamiento o una mezcla de ambos (ver la Ilustración 4).

- Placa de Nodos: Este módulos es un conjunto de nodos que son localmente organizados según los procesos de comunicación. Por esta razón la placa de nodos incluye un switch local que conecta todos sus nodos, actuando como un único puerto de comunicación. El número de nodos y las características del switch son totalmente personalizables por el usuario.

- Rack: Este módulo en un conjunto de varias placas de nodos utilizado con los fines de gestión y configuración. Cada placa de nodos en el rack posee su propio canal de comunicación. El número de placas de nodos es totalmente personalizable por el usuario.

La Ilustración 3 muestra los conjuntos básicos descritos anteriormente. El resto de la arquitectura (principalmente nodos de almacenamiento y el resto de switches de comunicación) puede ser agrupada en diferentes conjuntos. Además, el conjunto de módulos no está limitado únicamente a los módulos expuestos en esta sección. Se pueden definir nuevos módulos y añadirlos al repositorio de SIMCAN aumentando así el ámbito de configuraciones posibles para desarrollar nuevas arquitecturas y entornos.

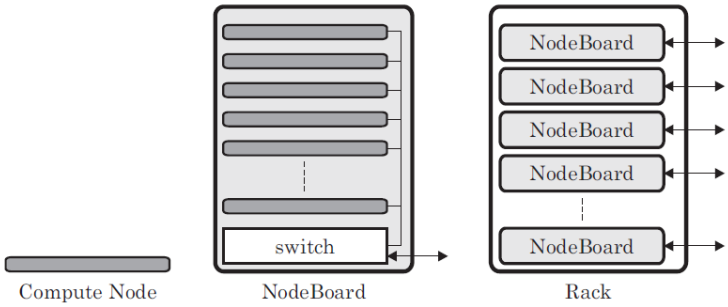

Ilustración 3 Paquetes SIMCAN

En un sistema de computación el nodo es el componente más importante. Igualmente, en SIMCAN un nodo es un bloque de construcción para crear sistemas distribuidos. En general un entorno distribuido está formado por nodos, dispositivos de comunicación como routers o switches, y redes de comunicaciones. Además, la arquitectura de SIMAN es flexible y no se restringe únicamente al uso de los nodos existentes para la construcción de entornos. En un nodo SIMCAN los cuatro sistemas básicos y todas las aplicaciones están conectados a la API del módulo (ver Ilustración 4).

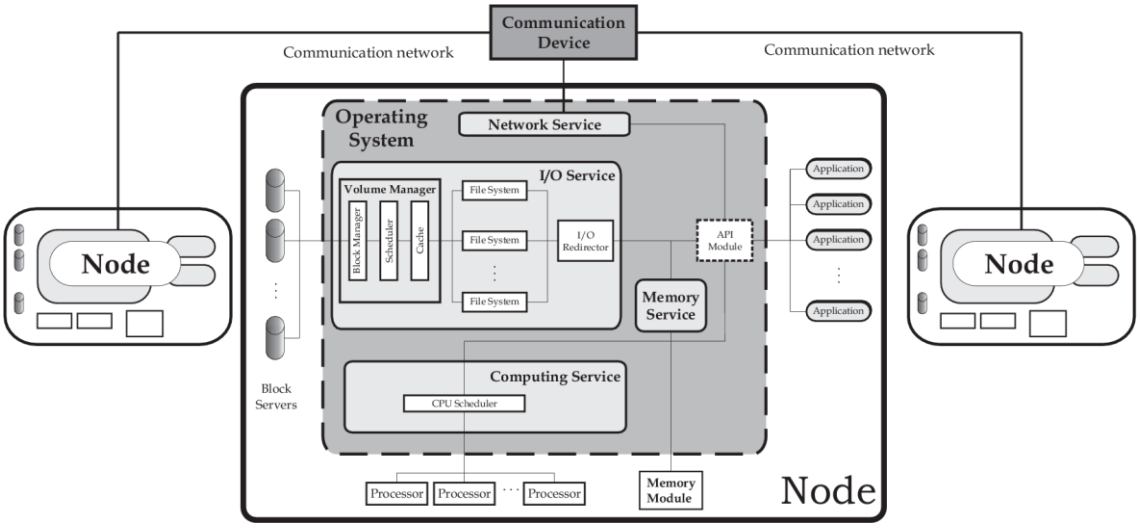

Ilustración 4: Arquitectura global de SIMCAN

3.3 Características

Las características más notables de la plataforma de simulación SIMCAN son las siguientes:

- Las arquitecturas existentes e inexistentes pueden ser modeladas y simuladas.
- Provee una API basada en POSIX para simular nuevas aplicaciones.
- Equilibra la relación entre escalabilidad, rendimiento y precisión para simular rápidamente entornos a gran escala con un nivel de detalle razonable.
- Permite la partición automática de modelos a gran escala para realizar una simulación paralela.
- Un nodo puede ser totalmente personalizado mediante la definición de cada sistema independientemente, lo que incluye:
    1. Sistema de computación
    2. Sistema de memoria.
    3. Sistema de almacenamiento
    4. Sistema de redes.
- El sistema de computación puede simular rápidamente sistemas multi proceso o de un proceso mediante distintas políticas de planificación.
- El sistema de memoria es muy útil para enumerar la cantidad de memoria necesaria por las aplicaciones para distintos usos, como: código, variables locales y globales, variables dinámicas y disco cache.

- Contiene un sistema de almacenamiento muy detallado, que incluye modelos para:

    1. Modelar sistemas de ficheros de uso general como Ext2 y Reiser FS.

    2. Modelar sistema de ficheros paralelos como PVFS.

    3. Modelar unidades de disco duro de uso general.

    4. Modelar sistemas RAID

    5. Modelar sistemas de ficheros remotos como NFS.

- El sistema de redes puede ser modelado para simular un amplio espectro de entornos distribuidos mediante el uso de distintos niveles de detalle:

    1. Nivel bajo de detalle, donde cada salto es representado usando su propia latencia, ancho de banda y tiempo de procesamiento.

    2. Nivel alto de detalle, usando el *framework* INET que también permite modelar el protocolo de pilas y las colisiones de datos.

- La misma arquitectura puede ser modelada con distintos niveles de detalle.

- Permite varios métodos para simular aplicaciones como

    1. Usando trazas de aplicaciones reales.

    2. Usando gráficos de estado.

    3. Programando aplicaciones nuevas directamente en la plataforma SIMCAN.

- Provee de un adaptador de la librería MPI para modelar y simular aplicaciones MPI.
- Incluye un amplio rango de componentes para construir y personalizar una amplia variedad de arquitecturas.
- Se puede añadir nuevos componentes al repositorio de SIMCAN.
- Los simuladores existentes y los nuevos se pueden integrar con la plataforma de simulación con el fin de simular partes concretas de una arquitectura modelada.

4 ANÁLISIS

En este capítulo se va a realizar un análisis del sistema que se pretende desarrollar. Para ello, y tras una breve introducción, se comenzará con una descripción de la aplicación, de la cual se obtendrán una serie de requisitos de usuario que deberá cumplir el sistema y un modelo inicial a partir del cual se podrá trabajar en el posterior diseño del mismo. Finalmente, se va a describir la funcionalidad del sistema y los requisitos necesarios para el desarrollo completo de la aplicación.

4.1 Introducción

La aplicación a desarrollar va a guiar al usuario en la creación y configuración de entornos de sistemas distribuidos a gran escala. Se basará en la plataforma de simulación SIMCAN descrita en el capítulo anterior.

El usuario tendrá la posibilidad de utilizar la versión de la plataforma SIMCAN que tenga en su unidad local, o podrá importar el fichero para obtener los datos de parametrización de SIMCAN.

Una vez cargado el fichero con la información de SIMCAN el usuario tendrá la posibilidad de configurar el escenario y sus elementos mediante una interfaz gráfica. El usuario será guiado tanto en la creación y modificación de los elementos que componen el escenario, como en las conexiones entre los mismos.

Con el fin de guiar al usuario la aplicación no permitirá operaciones que conlleven a errores o a la pérdida de coherencia entre valores utilizados en distintas partes de la configuración del escenario. Tampoco permitirá la inserción de datos que contengan valores de datos no válidos.

4.2 Descripción de la aplicación

Para este proyecto se define como escenario la representación de un entorno de distribución a gran escala basado en la plataforma de simulación SIMCAN. Un escenario puede estar compuesto por los siguientes elementos descritos con anterioridad y que son:

- Nodo
- Switch
- Rack

La configuración del escenario consta de la creación de los elementos individualmente y de su relación con el conjunto. Se puede resumir en 2 fases diferenciadas:

1. Creación e inserción de los elementos
2. Conexión entre los elementos.

Para explicar a fondo el análisis del presente proyecto, se dividirá en tres apartados con el fin de abordar todos los aspectos que lo componen de manera detallada:

- Analizador sintáctico (Parser).
- Elementos del escenario.
- Funcionalidad a desarrollar.

El usuario tendrá a su disposición una interfaz gráfica que le permitirá la configuración de un entorno de simulación. Todos los parámetros a configurar por parte del usuario son los que están definidos en la plataforma de simulación SIMCAN.

Los parámetros necesarios para la configuración de los entornos de sistemas distribuidos se han de obtener de forma estructurada y jerárquica para poder mostrarse posteriormente. Por esta razón se ha de realizar un analizador sintáctico (parser) que obtenga la información necesaria del repositorio de la plataforma de simulación SIMCAN.

4.3 Elementos del escenario.

Con el objetivo de configurar entornos de simulación, el usuario ha de poder utilizar y editar los siguientes elementos:

1. Nodos
2. Switches
3. Racks

4.3.1 Nodos.

Los nodos son los elementos más complejos que el usuario dispondrá para configurar. Se caracterizan por tener un elevado número de parámetros distribuidos entre los distintos niveles que lo componen. Se estructuran de la siguiente manera:

- Sistema operativo
- CPU
- Block Server
- Aplicaciones
- Si es un nodo de almacenamiento.

Configuración del Sistema operativo

La configuración del sistema operativo se compone de parámetros definidos en la plataforma de simulación SIMCAN. Al igual que en el resto de elementos, dichos parámetros se han de generar dinámicamente según están definidos en la plataforma SIMCAN, para mostrarse finalmente al usuario por medio de la interfaz gráfica de la aplicación.

Sin embargo, para la configuración del sistema operativo existe una parte que no obtiene todos los parámetros de la plataforma SIMCAN. Esta parte es fija y se compone de los siguientes elementos:

- File System
- Volume Manager

El sistema de ficheros (File system) no obtendrá datos del fichero que contiene la información de la plataforma SIMCAN. Se compondrá de una tabla que contendrá los siguientes parámetros: tipo de sistema de fichero, número de bloques, tamaño de bloques, índice de llenado, fichero de precarga y el destino del VFS. Tendrá tantas filas como números de sistemas de ficheros se hayan definido en el nivel superior, en el sistema operativo.

El administrador de volúmenes (volume manager) se definirá con una parte fija, que se corresponderá a las variables definidas en el nivel superior (sistema operativo), que son: el número de sistema de ficheros y el número de servidores de bloque. El resto de parámetros se obtendrán dinámicamente de la plataforma SIMCAN.

La configuración del sistema operativo se compondrá, además de los elementos ya expuestos, de parámetros que se definen según otro módulo de la plataforma SIMCAN, denominados submódulos. Esta es la manera de expresar

que un módulo contiene a otros. La configuración de este tipo de parámetros se llevará a cabo según los siguientes pasos:

1. La elección del tipo. Pueden existir varios tipos para el mismo módulo. Éstos contienen sus propios parámetros de configuración.
2. La configuración de los parámetros en función del tipo seleccionado.

Para el primer paso, la aplicación deberá obtener todos los tipos existentes de módulos de la plataforma SIMCAN, correspondientes al especificado en el parámetro. El usuario elegirá entre los distintos elementos mostrados, aquel que desea configurar. De esta manera al usuario se le facilitará, en el momento, un listado con las distintas opciones de las que dispone. Una vez elegido el tipo de módulo, se mostrarán todos los parámetros configurables correspondientes al módulo seleccionado.

Como se explicó anteriormente, en el apartado de la obtención del fichero de configuración de la plataforma SIMCAN, en la definición de los submódulos puede venir especificado el valor que se le asignarán a los subparámetros (parámetros del submódulo). Ese valor definido no es configurable y el usuario no podrá configurar ese parámetro directamente. Ésta es una de las ventajas de automatizar la configuración de este tipo de entornos. De este modo, se preserva la coherencia de los datos entre todos los elementos y se evitan errores ocasionados por descuidos o por valores incoherentes en este ámbito.

Para el segundo paso, se mostrará al usuario los parámetros necesarios para configurar el submódulo. Para ello será necesario buscar el módulo seleccionado por el usuario en el fichero y generar dinámicamente los campos correspondientes a los parámetros del módulo.

La aplicación tendrá que comprobar que los valores introducidos por el usuario se corresponden con los tipos de datos especificados en la plataforma SIMCAN.

Si todos los datos son válidos, se guardará la información introducida por el usuario.

Configuración de la CPU

La configuración de la CPU se realizará de una forma análoga a la del sistema operativo, a excepción de que no existe ningún elemento fijo que requiera de una configuración distinta o especial. Todos los parámetros de configuración se obtendrán de la plataforma SIMCAN, y por esta razón se generarán todos los campos de manera dinámica.

En este apartado de la configuración del nodo, también pueden aparecer submódulos y siempre se procederá de la misma manera que lo detallado en la configuración del sistema operativo: se mostrarán los campos parametrizables del submódulo para que el usuario los rellene. Los datos se validarán para comprobar que se han completado con arreglo al tipo dispuesto en la plataforma SIMCAN. Si los datos introducidos son válidos se almacenarán.

Configuración del Block Server

La configuración del Block Server se realizará de manera dinámica. Todos los elementos se cargarán del fichero de configuración de la plataforma SIMCAN. El procedimiento será igual al de los casos anteriores, se mostrarán los parámetros para completar al usuario. Si el parámetro se corresponde con un submódulo se mostrará una lista para que el usuario elija, entre los distintos elementos. Los elementos de esta lista se obtendrán de la plataforma SIMCAN, que son los módulos que se corresponden con el tipo pedido.

Igual que en los casos anteriores, una vez seleccionado el tipo de elemento, se configurarán los parámetros de este submódulo.

Para poder guardar la configuración, se deberán rellenar todos los parámetros configurables y deberán pertenecer al tipo de dato definido en SIMCAN.

Configuración de las Aplicaciones

En la configuración del nodo se definirán el número de aplicaciones que contendrán. Se deberá introducir un valor para poder acceder a la configuración de la ventana de aplicaciones. Debido a que el número de aplicaciones será introducido por el usuario, las aplicaciones se generarán de manera dinámica. Cada aplicación sólo deberá ser rellenada con el tipo, no contendrá más parámetros a este nivel. Para ello, igual en los casos anteriores, se mostrará un listado con las aplicaciones definidas en la plataforma SIMCAN, donde el usuario deberá elegir una por aplicación.

A continuación se mostrarán todos los parámetros configurables correspondientes al tipo seleccionado. Para poder guardar la información introducida se deberán rellenar todos los campos y que se correspondan al tipo de dato definido en SIMCAN.

Este proceso se repetirá sucesivamente por cada aplicación, hasta completar todas las aplicaciones especificadas por el usuario.

Una vez completados todos los niveles de configuración del nodo, todos los valores introducidos han de ser válidos, se le dará la opción al usuario de almacenar el nodo con el nombre que desee en el repositorio de nodos. El nodo creado ha de aparecer disponible al usuario para poder utilizarlo en la configuración de los entornos de distribución. De esta manera los nodos que almacenen en el repositorio se listarán para su posterior uso.

Los nodos han de ser editables. El usuario tendrá la posibilidad de escoger un nodo creado y modificar los parámetros que estime oportunos. Una vez realizados los cambios el usuario podrá guardarlos en el nodo inicial o guardarlo con otro nombre, creando así un nodo nuevo.

Mediante esta posibilidad se habilita la opción de usar un nodo ya creado como plantilla para crear otro. Se facilita así la configuración de parámetros, y ayuda a realizar cambios sin tener que recordar todos los parámetros ya configurados y probados en otros nodos, partiendo de una base segura.

4.3.2 Switches

Otro elemento configurable, además del nodo es el switch. Un switch en un dispositivo digital que permite la interconexión de varios segmentos de red. De ésta forma se pueden conectar varias redes en una única red.

A diferencia del nodo no tendrá distintos niveles o apartados de configuración, se compondrá de una única parte que será fija. Por esta razón no se obtendrá de la plataforma SIMCAN de manera dinámica como en el caso del nodo. Los parámetros a configurar serán cuatro:

- Aging Time
- Processing Time
- Buffer Size
- High Watermark

4.3.3 Racks

El último elemento configurable es el rack. El rack se define, en el contexto de este proyecto, como la cabina donde están dispuestos un conjunto de nodos (que serán del mismo tipo), formando una única estructura. Al igual que en otros casos obtendrá los parámetros configurables de la plataforma SIMCAN, a excepción del parámetro tipo de nodo, que se añade de forma fija.

El tipo de nodo será un elemento tipo, al igual que otros elementos ya explicados anteriormente, con la salvedad que en vez de listar los elementos de

la plataforma SIMCAN, se obtendrán del repositorio de nodos. Por esta razón los racks necesitarán de nodos creados previamente para su configuración. Un rack se define por medio de los siguientes elementos que lo forman:

- Num Boards
- Nodes per Board
- Node Type

4.4 Funcionalidades a desarrollar.

En este apartado se van a describir todos y cada uno de los requisitos del sistema necesarios para el desarrollo de la aplicación:

Gestionar escenarios

- Nuevo escenario
- Guardar escenario
- Guardar escenario como
- Cargar escenario

Gestionar repositorio SIMCAN

- Escanear directorio
- Cargar fichero de repositorio
- Guardar fichero de repositorio

Gestionar nodos

- Crear nodo

- Editar nodo
- Eliminar nodo
- Añadir nodo al escenario
- Añadir n nodos al escenario

Gestionar switches

- Crear switch
- Editar switch
- Eliminar switch
- Añadir switch al escenario
- Añadir n switches al escenario

Gestionar racks

- Crear rack
- Editar rack
- Eliminar rack
- Añadir rack al escenario
- Añadir n racks al escenario

Gestionar elementos del escenario

- Añadir conexión
- Eliminar conexión
- Renombrar elemento
- Cambiar icono
- Eliminar elemento del escenario

Gestionar simulación

- Generar archivos de configuración
- Habilitar simulación paralela
- Lanzar simulación

Una de las características que se buscan para esta aplicación es la facilidad de uso, que resulte intuitiva, para que el usuario invierta su tiempo en la creación y configuración de escenarios y no en tareas del sistema.

Además, se pretende potenciar la personalización del sistema, que el usuario tenga la posibilidad de configurar múltiples facetas a su gusto, tales como imágenes, nombres, etc. En el caso que el usuario no quiera definirlos el sistema los asignará las características establecidas por defecto.

4.5 Descripción de requisitos

Bajo estas premisas se ha ideado la funcionalidad del sistema. A continuación se describen de manera detallada cada una de los requisitos que satisfagan la funcionalidad del sistema. Se comenzará por la descripción de las operaciones principales y a continuación de las más simples que forman parte de las anteriores.

Los principales requisitos del sistema son los siguientes:

IDENTIFICADOR:	*RP-001*	TÍTULO:	**Gestionar escenarios**
DESCRIPCIÓN:	La base del presente proyecto radica en la creación de escenarios para poder llevar a cabo simulaciones de sistemas distribuidos a gran escala. Los escenarios agrupan toda la información necesaria sobre un entorno a simular. Por esta razón el sistema debe dar la opción al usuario de gestionar dichos recursos de una manera flexible.		

Tabla 2 Funcionalidad gestionar escenarios

IDENTIFICADOR:	*RP-002*	TÍTULO:	**Gestionar repositorio SIMCAN**
DESCRIPCIÓN:	La aplicación basará la configuración de los elementos del escenario a partir de la información contenida en el repositorio SIMCAN. El repositorio es susceptible de sufrir actualizaciones y modificaciones, por esta razón se ha de dar la opción al usuario de gestionar los ficheros de opciones que se obtienen de dicho repositorio.		

Tabla 3 Funcionalidad gestionar repositorio SIMCAN

IDENTIFICADOR:	*RP-003*	TÍTULO:	**Gestionar nodos**
DESCRIPCIÓN:	Los nodos son uno de los tres tipos de elementos que pueden formar parte de un escenario. Para poder ser utilizados, previamente se han de configurar según los tipos de parámetros definidos en la plataforma de simulación SIMCAN. Se almacenarán en un repositorio de elementos para poder ser añadidos a los distintos escenarios.		

Tabla 4 Funcionalidad gestionar nodos

IDENTIFICADOR:	*RP-004*	TÍTULO:	**Gestionar switches**
DESCRIPCIÓN:	Los switches son otro tipo de elemento que se pueden añadir a un escenario. Al igual que los nodos dispondrá de un repositorio donde se almacenarán después de ser correctamente configurados (por el tipo de dato no por el contenido), para finalmente, ser utilizados en los escenarios que se creen.		

Tabla 5 Funcionalidad gestionar switches

IDENTIFICADOR:	*RP-005*	TÍTULO:	**Gestionar racks**
DESCRIPCIÓN:	Los racks son otro tipo de elemento que se pueden añadir a un escenario. Al igual que los nodos y los switches dispondrá de un repositorio donde se almacenarán después de ser correctamente configurados (por el tipo de dato no por el contenido), para finalmente, ser utilizados en los escenarios que se creen.		

Tabla 6 Funcionalidad gestionar racks

IDENTIFICADOR:	*RP-006*	TÍTULO:	**Gestionar elementos**
DESCRIPCIÓN:	Los elementos que forman un escenario ofrecerán distintas posibilidades de configuración en dicho contexto. Los nombres serán únicos, dado que puede haber varios elementos del mismo tipo en el mismo escenario, y así poder diferenciarse. Se ofrecerán distintas opciones para cada elemento en aras de crear y editar escenarios de una manera intuitiva, como en la apariencia del propio elemento.		

Tabla 7 Funcionalidad gestionar elementos del escenario

IDENTIFICADOR:	*RP-007*	TÍTULO:	**Gestionar simulación**
DESCRIPCIÓN:	La finalidad de la aplicación reside en la creación de escenarios para su posterior simulación. Para llevar a cabo la simulación es necesaria la creación de los ficheros de salida, que contendrán la información del escenario creado según el formato del simulador *OMNeT++*. Finalmente se le facilitará al usuario el lanzamiento de la simulación desde la propia aplicación, siempre y cuando esté correctamente configurada esta opción.		

Tabla 8 Funcionalidad gestionar simulación

Como producto de la simplificación de las distintas funcionalidades comentadas anteriormente y teniendo como objetivo conseguir un mayor grado de detalle en ellas, surgen una serie de funcionalidades básicas, derivadas de las anteriores, que debe cumplir la aplicación. Estas funcionalidades básicas son las siguientes:

IDENTIFICADOR:	*RB-001*	TÍTULO:	**Añadir conexión**
DESCRIPCIÓN:	Los elementos que componen un escenario se interrelacionarán por medio de conexiones. Las conexiones siempre se llevarán a cabo por medio del elemento switch, debido a que es el único elemento que permite la interconexión de diferentes elementos. Por esta razón, los elementos nodo y rack no podrán relacionarse directamente entre ellos, solamente se podrán conectar a un único elemento switch. Sin embargo un switch puede conectarse a cualquier elemento incluyendo otros switches.		

Tabla 9 Funcionalidad añadir conexión

IDENTIFICADOR:	*RB-002*	TÍTULO:	**Eliminar conexión**
DESCRIPCIÓN:	El sistema debe permitir la eliminación de conexiones entre un elemento switch y otro elemento del escenario realizadas por el usuario. La aplicación deberá pedir confirmación al usuario sobre el borrado y en el caso de que la contestación sea afirmativa, deberá eliminar la conexión que atañe a los dos elementos. Debido a que un elemento swtich puede tener múltiples conexiones se creará un gestor para habilitar esta funcionalidad.		

Tabla 10 Funcionalidad eliminar conexión

IDENTIFICADOR:	*RB-003*	TÍTULO:	**Renombrar elemento**
DESCRIPCIÓN:	Cada vez que se añade un nodo al escenario, el sistema establece un nombre único en el contexto del escenario. Dado que existe la posibilidad de añadir al escenario varios nodos del mismo tipo (los configurados por el usuario) el sistema construirá el nombre en función del tipo de elemento (nodo, switch, rack) y un número. El usuario tiene la posibilidad de modificar el nombre y establecer otro a su elección, siempre y cuando no esté siendo utilizado en el escenario. Si se da la situación que el nombre ya existe le aparecerá un mensaje informativo indicando que el nombre ya está en uso. Si el nombre no está siendo utilizado en el escenario, se actualizará.		

Tabla 11 Funcionalidad renombrar elemento

IDENTIFICADOR:	*RB-004*	TÍTULO:	**Cambiar icono**
DESCRIPCIÓN:	Con objeto de hacer la interfaz lo más agradable posible, cada vez que se añade un elemento nuevo al escenario se le asigna una imagen por defecto. La imagen preestablecida será distinta según el tipo de elemento, y así poder ser diferenciados a primera vista. Para poder modificar la imagen asociada a un elemento, se deberá seleccionar y elegir la imagen a cambiar, que puede encontrarse en cualquier carpeta del sistema. Para facilitar la elección se previsualizarán las imágenes seleccionadas antes de aceptar la modificación.		

Tabla 12 Funcionalidad cambiar icono

IDENTIFICADOR:	*RB-005*	TÍTULO:	**Eliminar elemento**
DESCRIPCIÓN:	El sistema permitirá el borrado de los elementos añadidos al escenario. A su vez se borrarán automáticamente todas las conexiones que tuviera dicho elemento. Para ello el usuario deberá seleccionar el elemento a borrar y elegir la opción requerida. Si contiene conexiones se eliminarán actualizando los elementos afectados. Finalmente se mostrará al usuario la desaparición del elemento y la actualización de las conexiones a la nueva situación.		

Tabla 13 Funcionalidad eliminar elemento del escenario

IDENTIFICADOR:	*RB-006*	TÍTULO:	**Escanear directorio**
DESCRIPCIÓN:	El presente proyecto se basa en la plataforma de simulación SIMCAN para la configuración de los elementos que componen un escenario. Para poder incorporar la información de SIMCAN, la aplicación proporcionará la funcionalidad de crear un archivo de configuración que contenga todas las especificaciones de la plataforma SIMCAN relevantes para el sistema. El usuario determinará el directorio donde se encuentra el repositorio SIMCAN y el sistema recorrerá todos los archivos que lo forman obteniendo la información útil para el sistema. Se generará un archivo de configuración (también descrito como archivo de repositorio) que establecerá los parámetros configurables para los elementos que forman un escenario.		

Tabla 14 Funcionalidad escanear directorio

IDENTIFICADOR:	*RB-007*	TÍTULO:	**Guardar fichero de repositorio**
DESCRIPCIÓN:	El sistema ofrece la opción de cargar el fichero de repositorio, de la misma manera ha de permitir la exportación del fichero que esté cargado en ese momento. El fichero cargado será el escaneado, si el usuario ha llevado a cabo esa opción, o uno cargado del repositorio. Los ficheros escaneados se almacenan en un fichero de trabajo por defecto. Al realizar un nuevo escaneo se sobrescribirá, para almacenar la versión o para poder utilizarlo en otro equipo, se ofrecerá la opción de guardar el fichero de repositorio.		

Tabla 15 Guardar fichero de repositorio

IDENTIFICADOR:	*RB-008*	TÍTULO:	**Cargar fichero de repositorio**
DESCRIPCIÓN:	El sistema ofrecerá la posibilidad al usuario de cargar directamente un archivo de configuración de la plataforma SIMCAN sin necesidad de escanearlo del propio ordenador. De esta manera se dotará de más flexibilidad a la aplicación, que permitirá la utilización de diferentes versiones favoreciendo el uso entre distintos usuarios o lugares. En determinadas ocasiones será necesaria la opción de cargar un fichero de repositorio, bien porque no se encuentre ningún repositorio de la plataforma SIMCAN en el PC o porque se desee cargar otra versión.		

Tabla 16 Funcionalidad cargar fichero de repositorio

IDENTIFICADOR:	*RB-009*	TÍTULO:	**Crear nodo**
DESCRIPCIÓN:	La aplicación requerirá de elementos creados para poder llevarse a cabo la configuración de un escenario. Una vez creados podrán añadirse al configurador para definir el escenario que representa un sistema distribuido. En la zona de la gestión de nodos, el usuario tendrá la opción de elegir la función crear nodo. Una vez seleccionado aparecerá el configurador de nodos, descrito anteriormente en esta sección, que guiará al usuario en la parametrización de dicho elemento.		

Tabla 17 Funcionalidad crear nodo

IDENTIFICADOR:	*RB-010*	TÍTULO:	**Editar nodo**
DESCRIPCIÓN:	Los nodos se han de poder editar, bien por cambios que el usuario quiere ajustar, o por una actualización del fichero de configuración que añada nuevas características a los nodos previamente definidos. Para poder realizar la modificación se ha de seleccionar en la zona de gestión de nodos el elemento requerido y elegir la opción editar nodo. Si se modifica un parámetro general habrá que volver a configurar los apartados que se vieran afectados para preservar la coherencia del nodo. La opción editar nodo también ha de servir para usar de plantilla un nodo existente y de esta manera modificar las características del nodo original conservando los dos elementos. Al guardarlo con otro nombre se creará un nodo nuevo y no será necesario partir de cero en la parametrización del elemento.		

Tabla 18 Funcionalidad editar nodo

IDENTIFICADOR:	*RB-011*	TÍTULO:	**Eliminar nodo**
DESCRIPCIÓN:	La aplicación permitirá el borrado de nodos almacenados. Cuando el usuario no quiera disponer más de un nodo tendrá que ir a la zona de gestión de nodos, seleccionar el elemento requerido y elegir la opción eliminar nodo. Esta opción no eliminará los nodos del tipo borrado que se hayan añadido al escenario. Si se desean eliminar se ha de proceder como se explicó en las funcionalidades del escenario y seleccionar la opción eliminar elemento del escenario por cada nodo que se desee borrar.		

Tabla 19 Funcionalidad eliminar nodo

IDENTIFICADOR:	*RB-012*	TÍTULO:	**Añadir nodo al escenario**
DESCRIPCIÓN:	Una vez que existen nodos creados en el repositorio, el sistema dará la opción de seleccionar un elemento y añadirlo al escenario activo. Le asignará un nombre de nodo por defecto, al igual que una imagen asociada a dicho elemento. Para poder aplicar un nombre o una imagen distinto, habrá que operar como se expuso en el punto de esta sección funcionalidades de escenario y llevar a cabo las funciones renombrar elemento o cambiar icono según convenga.		

Tabla 20 Funcionalidad añadir nodo al escenario

IDENTIFICADOR:	*RB-013*	TÍTULO:	**Añadir n nodos al escenario**
DESCRIPCIÓN:	Los sistemas distribuidos a gran escala requieren la configuración de un número muy elevado de elementos. Para posibilitar al usuario el uso masivo de elementos, se facilitará la opción de insertar directamente el número de elementos de un mismo tipo. Se añadirá esta opción en la zona de la gestión de nodos. El usuario seleccionará el nodo a añadir mediante la función añadir n nodos al escenario, y deberá introducir el número de elementos que quiere agregar al escenario activo.		

Tabla 21 Funcionalidad añadir n nodos al escenario

IDENTIFICADOR:	*RB-014*	TÍTULO:	**Añadir swtich**
DESCRIPCIÓN:	La aplicación requerirá de elementos creados para poder llevarse a cabo la configuración de un escenario. Una vez creados podrán añadirse al configurador para definir el escenario que representa un sistema distribuido. En la zona de la gestión de switches, el usuario tendrá la opción de elegir la función crear switch. Una vez seleccionado aparecerá el configurador de switches, descrito anteriormente en esta sección, que guiará al usuario en la parametrización de dicho elemento.		

Tabla 22 Funcionalidad añadir switch

IDENTIFICADOR:	*RB-015*	TÍTULO:	**Editar swtich**
DESCRIPCIÓN:	Los switches han de poder ser editados, bien por cambios que el usuario quiere ajustar, o por una actualización del fichero de configuración que añada nuevas características a los switches previamente definidos. Para poder realizar la modificación se ha de seleccionar en la zona de gestión de switches el elemento requerido y elegir la opción editar swtich. La opción editar switch también ha de servir para usar de plantilla un switch existente y de esta manera modificar las características del swtich inicial conservando los dos elementos. Al guardarlo con otro nombre se creará uno nuevo y no será necesario partir de cero en la parametrización del elemento.		

Tabla 23 Funcionalidad editar swtich

IDENTIFICADOR:	*RB-016*	TÍTULO:	**Eliminar swtich**
DESCRIPCIÓN:	La aplicación permitirá el borrado de switches almacenados. Cuando el usuario no quiera disponer más de un swtich, tendrá que ir a la zona de gestión de switches, seleccionar el elemento requerido y elegir la opción eliminar switch. Esta opción no eliminará los switches del tipo borrado que se hayan añadido al escenario. Si se desean eliminar se ha de proceder como se explicó en las funcionalidades del escenario y seleccionar la opción eliminar elemento del escenario por cada switch que se desee borrar.		

Tabla 24 Funcionalidad eliminar switch

IDENTIFICADOR:	*RB-017*	TÍTULO:	**Añadir swtich al escenario**
DESCRIPCIÓN:	Una vez que existen switches creados en el repositorio, el sistema dará la opción de seleccionar un elemento y añadirlo al escenario activo. Le asignará un nombre al switch por defecto, al igual que una imagen asociada a dicho elemento. Para poder aplicar un nombre o una imagen distinto, habrá que operar como se expuso en el punto de esta sección funcionalidades de escenario y llevar a cabo las funciones renombrar elemento o cambiar icono según convenga.		

Tabla 25 Funcionalidad añadir switch al escenario

IDENTIFICADOR:	*RB-018*	TÍTULO:	**Añadir n switches al escenario**
DESCRIPCIÓN:	Los sistemas distribuidos a gran escala requieren la configuración de un número muy elevado de elementos. Para posibilitar al usuario el uso masivo de elementos, se facilitará la opción de insertar directamente el número de elementos de un mismo tipo. Se añadirá esta opción en la zona de la gestión de switches. El usuario seleccionará el swtich a añadir mediante la función añadir n switches al escenario, y deberá introducir el número de elementos que quiere agregar al escenario activo.		

Tabla 26 Funcionalidad añadir n switches al escenario

IDENTIFICADOR:	*RB-019*	TÍTULO:	**Crear rack**
DESCRIPCIÓN:	La aplicación requiere de elementos creados para poder llevarse a cabo la configuración de un escenario. Una vez creados podrán añadirse al configurador para definir el escenario que representa un sistema distribuido. En la zona de la gestión de racks, el usuario tendrá la opción de elegir la función crear rack. Una vez seleccionado aparecerá el configurador de racks, descrito anteriormente en esta sección, que guiará al usuario en la parametrización de dicho elemento.		

Tabla 27 Funcionalidad crear rack

IDENTIFICADOR:	*RB-020*	TÍTULO:	**Editar rack**
DESCRIPCIÓN:	Los racks han de poder ser editados, bien por cambios que el usuario busca ajustar, o por una actualización del fichero de configuración que añada nuevas características a los racks previamente definidos. Para poder realizar la modificación se ha de seleccionar en la zona de gestión de racks el elemento requerido y elegir la opción editar rack. La opción editar rack también ha de servir para usar de plantilla un rack existente y de esta manera modificar las características del rack inicial conservando los dos elementos Al guardarlo con otro nombre se creará un rack nuevo y no será necesario partir de cero en la parametrización del elemento.		

Tabla 28 Funcionalidad editar rack

IDENTIFICADOR:	*RB-021*	TÍTULO:	**Eliminar rack**
DESCRIPCIÓN:	La aplicación permitirá el borrado de los racks almacenados. Cuando el usuario no quiera disponer más de un rack tendrá que ir a la zona de gestión de racks, seleccionar el elemento requerido y elegir la opción eliminar rack. Esta opción no eliminará los racks del tipo borrado que se hayan añadido al escenario. Si se desean eliminar se ha de proceder como se explicó en las funcionalidades del escenario y seleccionar la opción eliminar elemento del escenario por cada rack que se desee borrar.		

Tabla 29 Funcionalidad eliminar rack

IDENTIFICADOR:	*RB-022*	TÍTULO:	**Añadir rack al escenario**
DESCRIPCIÓN:	Una vez que existen racks creados en el repositorio, el sistema dará la opción de seleccionar un elemento y añadirlo al escenario activo. Le asignará un nombre de rack por defecto, al igual que una imagen asociada al elemento. Para poder aplicar un nombre o una imagen distinto, habrá que operar como se expuso en el punto de esta sección funcionalidades de escenario y llevar a cabo las funciones renombrar elemento o cambiar icono según convenga.		

Tabla 30 Funcionalidad añadir rack al escenario

IDENTIFICADOR:	*RB-023*	TÍTULO:	**Añadir n racks al escenario**
DESCRIPCIÓN:	Los sistemas distribuidos a gran escala requieren la configuración de un número muy elevado de elementos. Para posibilitar al usuario el uso masivo de elementos, se facilitará la opción de insertar directamente el número de elementos de un mismo tipo. Se añadirá esta opción en la zona de la gestión de racks. El usuario seleccionará el rack a añadir mediante la función añadir n racks al escenario, y deberá introducir el número de elementos que quiere agregar al escenario activo.		

Tabla 31 Funcionalidad añadir n racks al escenario

IDENTIFICADOR:	*RB-024*	TÍTULO:	**Nuevo escenario**
DESCRIPCIÓN:	Se define como escenario, en el contexto de este proyecto, al conjunto de elementos (nodos, racks y switches) y la interrelación entre ellos para formar un sistema. La aplicación deberá permitir la creación de un escenario desde cero, donde el usuario personalice según sus preferencias mediante las funcionalidades de escenario descritas anteriormente en esta sección de análisis.		

Tabla 32 Funcionalidad nuevo escenario

IDENTIFICADOR:	*RB-025*	TÍTULO:	**Guardar escenario**
DESCRIPCIÓN:	La aplicación deberá proporcionar al usuario la funcionalidad de almacenar el escenario activo para poder utilizarlo o modificarlo posteriormente. Se almacenarán todos los componentes que forman el escenario. Al guardar el escenario se deberán almacenar todas las conexiones existentes y la disposición de los elementos sobre el escenario.		

Tabla 33 Funcionalidad guardar escenario

IDENTIFICADOR:	*RB-026*	TÍTULO:	**Guardar escenario como**
DESCRIPCIÓN:	El sistema ofrecerá al usuario la opción de establecer el nombre del escenario en uso que desee. Podrá realizarlo en cualquier momento que elija guardar el escenario. Esta funcionalidad es exactamente igual a la anterior a excepción que pedirá al usuario que establezca un nombre para el escenario, mientras que de la otra manera que lo almacena con el nombre que ya tuviera establecido en la creación del escenario.		

Tabla 34 Funcionalidad guardar escenario como

IDENTIFICADOR:	*RB-027*	TÍTULO:	**Cargar escenario**
DESCRIPCIÓN:	La aplicación deberá proporcionar al usuario la opción de poder cargar un escenario previamente guardado. El usuario podrá tener múltiples escenarios almacenados pero sólo podrá utilizar uno, sólo habrá uno activo. Se cargarán todos los elementos, las conexiones establecidas y se mostrará la disposición de los elementos sobre el escenario de la misma forma que cuando fue guardado por última vez por parte del usuario.		

Tabla 35 Funcionalidad escenario

IDENTIFICADOR:	*RB-028*	TÍTULO:	**Generar archivos de configuración**
DESCRIPCIÓN:	El simulador *OMNeT*++requiere de dos ficheros que contengan la información necesaria para realizar la simulación de un entorno distribuido. Los ficheros son: • Omnet.ini: El fichero omnet.ini es aquel que va a contener toda la información referente a los elementos que forman el escenario, y también describe la configuración general de la simulación. • Scenario.ned: El fichero scenario.ned dispondrá de la información requerida por el simulador OMNET para especificar la topología de red. Cuando un escenario esté cargado en la aplicación podrá seleccionarse la opción generar ficheros y se crearán en directorio correspondiente al escenario abierto.		

Tabla 36 Funcionalidad generar archivos de configuración

IDENTIFICADOR:	*RB-029*	TÍTULO:	**Habilitar simulación paralela**
DESCRIPCIÓN:	La simulación en *OMNeT*++se puede llevar a cabo de manera secuencial o de manera paralela. Habilitando o deshabilitando esta opción el usuario establece la forma en la que desea que se realice, y de esta forma se escribirá en los ficheros de configuración.		

Tabla 37 Funcionalidad habilitar simulación paralela

IDENTIFICADOR:	*RB-030*	TÍTULO:	**Lanzar simulación**
DESCRIPCIÓN:	Cuando se realiza un número elevado de pruebas, las tareas repetitivas llevan al usuario a consumir mucho tiempo en pequeñas acciones. Para evitar esta situación y con el objetivo de hacer más útil la aplicación, se añadirá la funcionalidad de lanzar la simulación desde la propia aplicación. De esta manera no será necesario introducir los datos relativos a la configuración del escenario en el terminal para ejecutar el simulador *OMNeT++*. Cada vez que se quiera lanzar una simulación, se llevará a cabo automáticamente si el usuario selecciona esta función.		

Tabla 38 Funcionalidad lanzar simulación

4.6 Casos de Uso

Una vez expuestas las funcionalidades a implementar, se puede realizar un diagrama general de casos de uso para representar la interacción del sistema con el usuario (o con los actores, como se denominan en este tipo de diagramas). Con el fin de conseguir una mejor visión de la información, se ha decidido dividir el esquema principal en una serie de subesquemas, cada uno de los cuáles muestra una parte de la funcionalidad antes descrita. Así, primero se muestran los casos de uso principales y posteriormente se describe cada uno de ellos junto con los casos derivados del mismo.

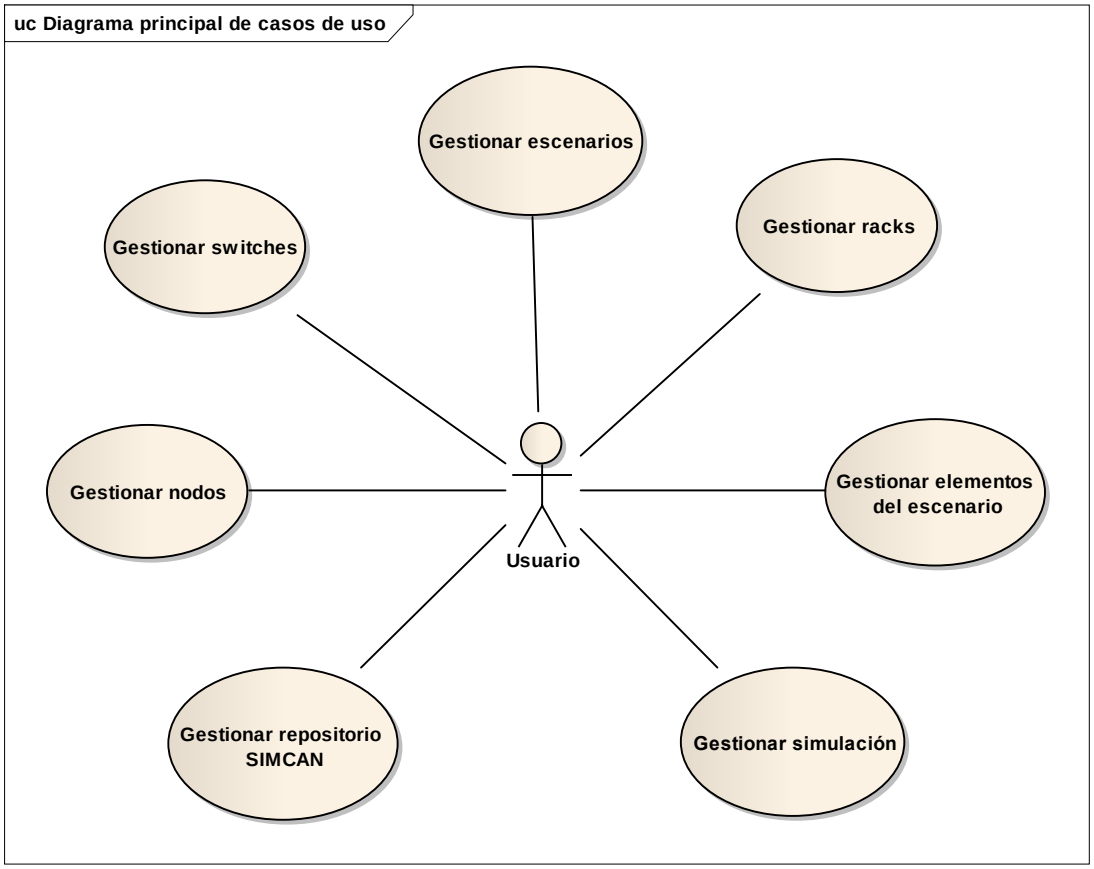

Ilustración 5 Casos de uso principales

En la Ilustración 5 se muestra el esquema principal de casos de uso del sistema. En él se muestran las posibles acciones entre los usuarios (el actor) y la aplicación. La funcionalidad principal definida a lo largo de este documento se divide en siete casos de uso principales: gestión de escenarios, gestión de nodos, gestión de racks, gestión de switches, gestión de elementos del escenario, gestión del repositorio SIMCAN y gestión de la simulación. Cada uno de estos casos de uso individuales se divide a su vez en una serie de casos de uso básicos.

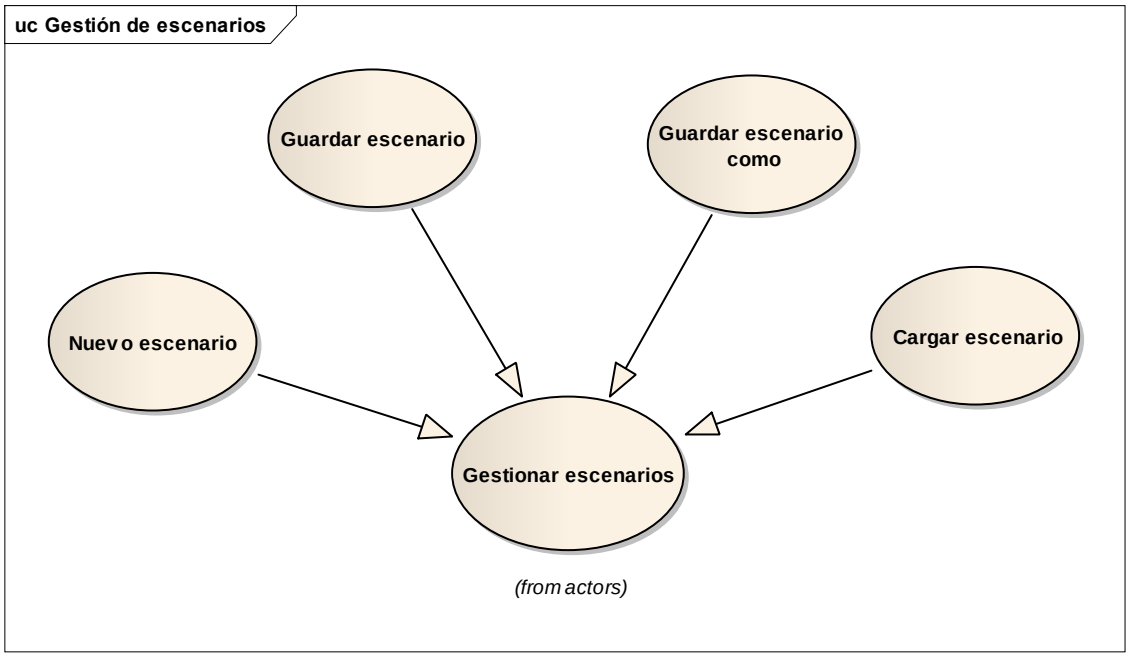

Ilustración 6 Casos de uso derivados de la gestión de escenarios

En la Ilustración 6 se muestran los casos de uso derivados de la gestión de escenarios. Se puede apreciar como esta funcionalidad se ha dividido en otras cuatro en función a las acciones a realizar.

En los siguientes casos de uso se detallan las funcionalidades posibles de los elementos del sistema a nivel de configuración. Como se ha expuesto con anterioridad, existen tres tipos de elementos: nodos, switches y racks. Cada tipo de elemento tendrá su propia gestión, no será genérica, y además contarán con su propio repositorio para poder organizar de una forma estructurada los elementos de la aplicación.

En la Ilustración 7 se exponen los casos de usos derivados de la gestión de nodos.

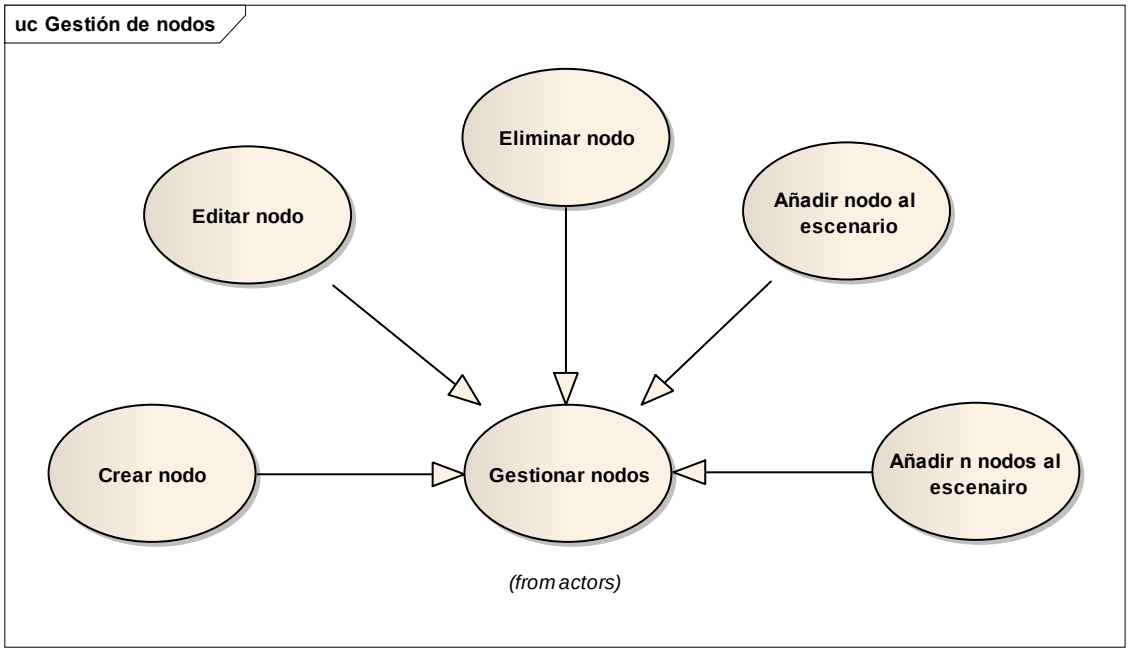

Ilustración 7 Casos de usos derivados de la gestión de los nodos

De manera análoga a la gestión de nodos, se definen los casos de uso derivados de la gestión de racks en la Ilustración 8, y los casos de uso derivados de la gestión de switches en la Ilustración 9:

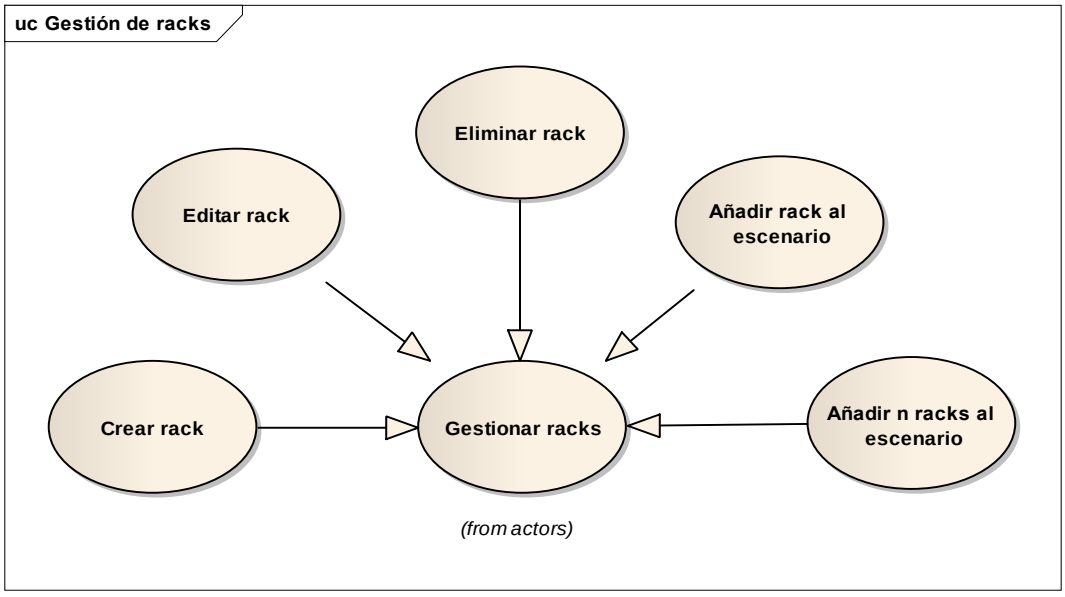

Ilustración 8 Casos de usos derivados de la gestión delos racks

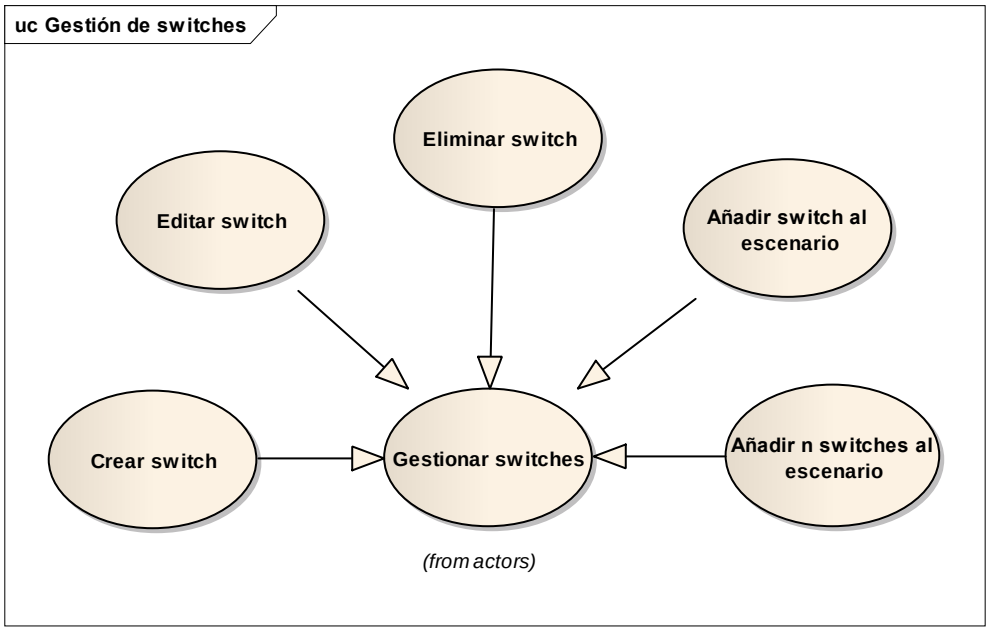

Ilustración 9 Casos de usos derivados de la gestión de los switches

Los casos de uso derivados de la gestión del repositorio SIMCAN, como se puede apreciar en la Ilustración 10 se dividen en tres casos:

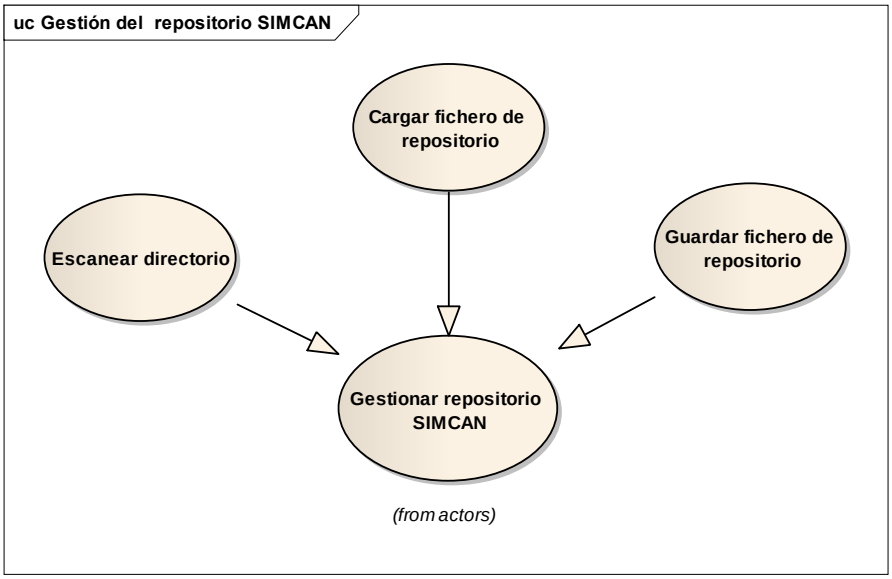

Ilustración 10 Casos de usos derivados de la gestión del repositorio SIMCAN

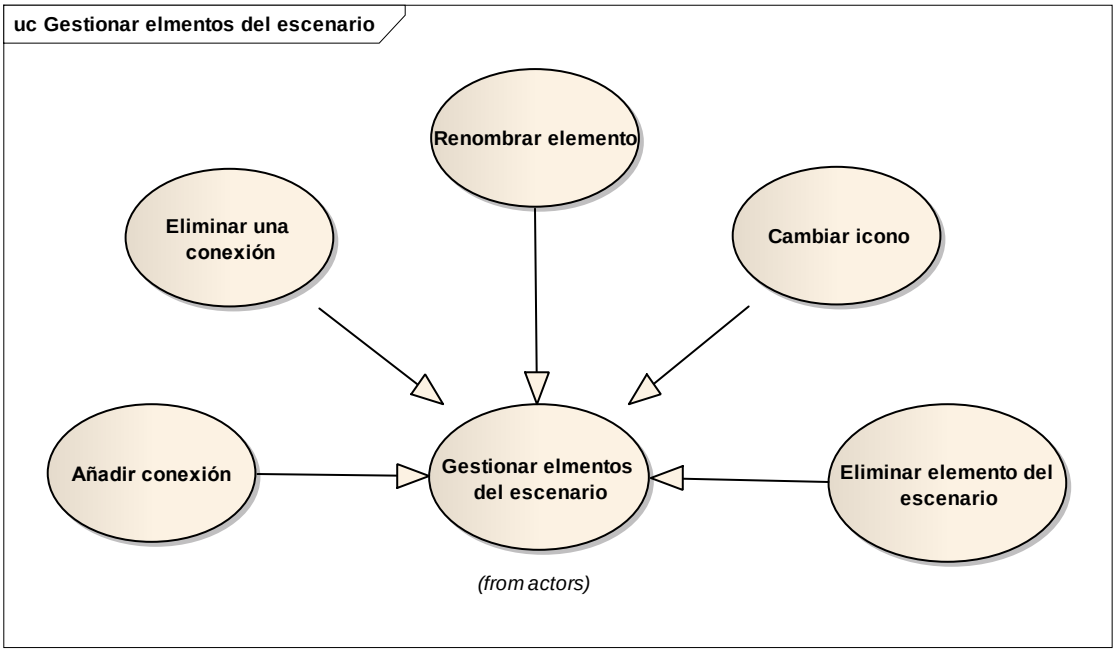

Ilustración 11 Casos de uso derivados de la gestión de los elementos del escenario

Con el objetivo de dotar al usuario de cierta capacidad de configuración de los elementos del sistema, para que el usuario pueda personalizar dichos elementos dentro de las posibilidades de la aplicación y hacer uso de una manera intuitiva de las mismas, se han definido la gestión de los elementos del escenario. Los casos de uso derivados de este caso principal se dividen en cinco para ofrecer las posibilidades descritas en la funcionalidad del sistema al respecto como se recoge en la Ilustración 11.

Finalmente se puede observar en la Ilustración 12 los casos de uso derivados de la gestión de la simulación, que a su vez son las funcionalidades que cierran el fin último de la aplicación.

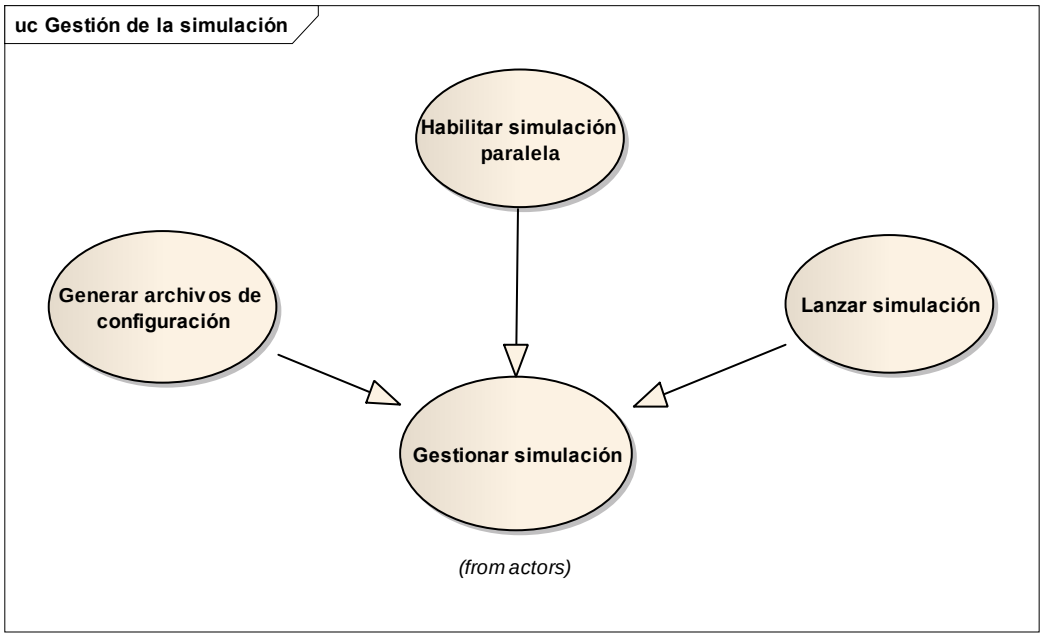

Ilustración 12 Casos de uso derivados de la gestión de la simulación

5 DISEÑO

En este apartado se expondrá el modelo lógico del sistema a partir de la información recogida en el análisis. Además se detallará toda la parte de interfaz que atañe a la interacción del usuario con la aplicación. Se explicará cómo se distribuye el flujo de información, y cómo se obtiene todos los datos de la plataforma SIMCAN a partir del analizador sintáctico. Seguidamente, se describirá de manera detallada cómo se almacenará la información una vez configurada por parte del usuario. Finalmente se explicarán las distintas partes en las que se dividen los ficheros que serán generados por la aplicación.

5.1 Introducción

Con objeto de clarificar los conceptos principales del sistema y en aras de facilitar la comprensión de los siguientes puntos que componen la sección del diseño, se procede a una breve introducción global del sistema.

La unidad que engloba un entorno donde se definen los componentes y sus conexiones se denomina escenario. El escenario es la base donde el usuario crea o modifica el entorno de simulación distribuido.

Para poder utilizar un escenario se necesita tener configurados, o configurarlos en ese momento, los elementos del sistema que cómo ya se expuso anteriormente son: nodo, rack y switch.

Posteriormente se detallará a fondo las distintas posibilidades de configuración que se ofrecen para cada uno de ellos y su rol y opciones dentro del propio escenario.

Sin embargo, todos los parámetros de configuración necesarios para cada elemento no están definidos en el sistema. Para poder obtenerlos se han de obtener de la plataforma de simulación SIMCAN. Dicha plataforma está compuesta por un repositorio de ficheros que definen los módulos repartidos en una estructura lógica.

Para poder inferir los datos necesarios se necesita un analizador sintáctico. Éste recorrerá todos los ficheros que contengan la información referente a su correspondiente módulo y finalmente lo almacenará en un fichero xml. El fichero xml tiene una jerarquía que se detallará, al igual que el funcionamiento del analizador, el en siguiente punto de esta sección: Analizador sintáctico.

En el escenario se podrán realizar conexiones entre los elementos por medio del elemento switch. El usuario podrá elegir el elemento origen y destino (uno de los dos ha de ser un swtich) para establecer la conexión como se detalla en el punto referente a la Interfaz de usuario.

Finalmente, una vez definido el escenario, el usuario podrá lanzar la ejecución de la simulación por medio de los ficheros que se generarán (scenario.ned y omnet.ini) que son los archivos que requiere el simulador OMNeT++.

5.2 Modelo Lógico

El modelo representado en la Ilustración 13, contiene los principales conceptos incluidos en la especificación de requisitos de usuario además de las estructuras de datos necesarias para asegurar el correcto funcionamiento del sistema.

Mediante el modelo lógico se pueden observar las relaciones entre las clases que forman el sistema al más alto nivel, que definen la estructura global de la aplicación.

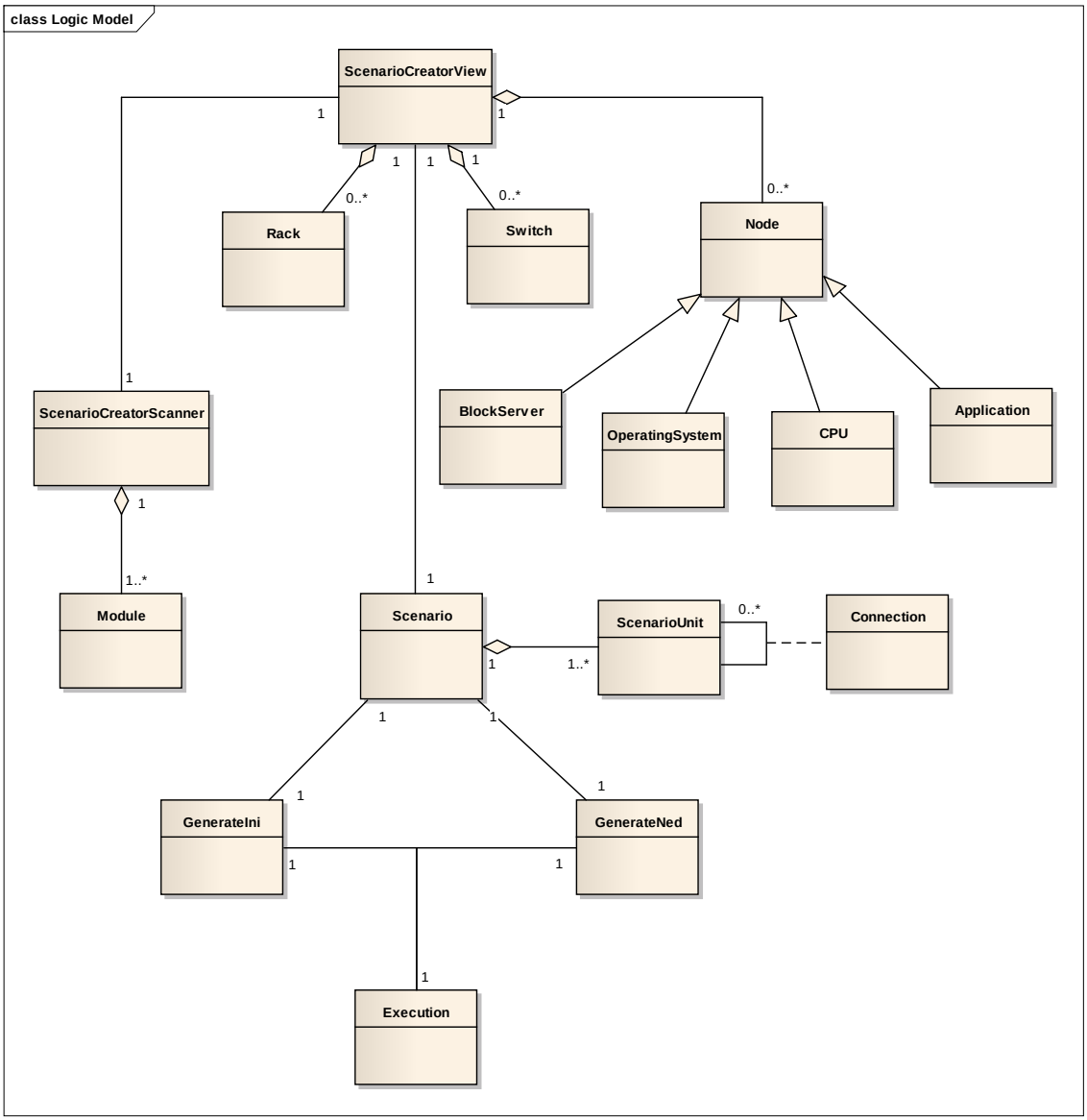

Ilustración 13 Modelo lógico del sistema

5.3 Flujo de información del sistema

Una vez que se han definido los requisitos del sistema a cubrir, se ha descrito la funcionalidad a implementar y se ha expuesto el modelo lógico es el momento de diseñar el flujo de información del sistema. Los puntos en los que se basa son los siguientes:

1. Se obtiene el fichero de configuración de los repositorios de la plataforma de simulación SIMCAN. Contendrá los parámetros de configuración de los distintos módulos que definen a la plataforma. En caso de que se hubiera obtenido con anterioridad, el fichero se detectará en el sistema y se cargará por defecto. Siempre será necesario tener al menos un fichero de configuración.

2. Deberán existir en la aplicación el sistema los elementos del sistema (nodos, switches y racks) que se quieran utilizar en el paso siguiente, la configuración del escenario. Para ello se deberán crear nuevos elementos o modificar los existentes.

3. La configuración del escenario la llevará a cabo el usuario, añadiendo los elementos del sistema previamente creados y estableciendo las conexiones entre los mismos, según sus especificaciones permitidas.

4. Para generar los ficheros para OMNeT++ deberá existir un escenario activo en la pantalla principal de configuración. En caso positivo se crearán los dos ficheros.

5. Una vez generados los ficheros, si el OMNeT++ está configurado en la máquina local, se podrá lanzar directamente la simulación desde la aplicación.

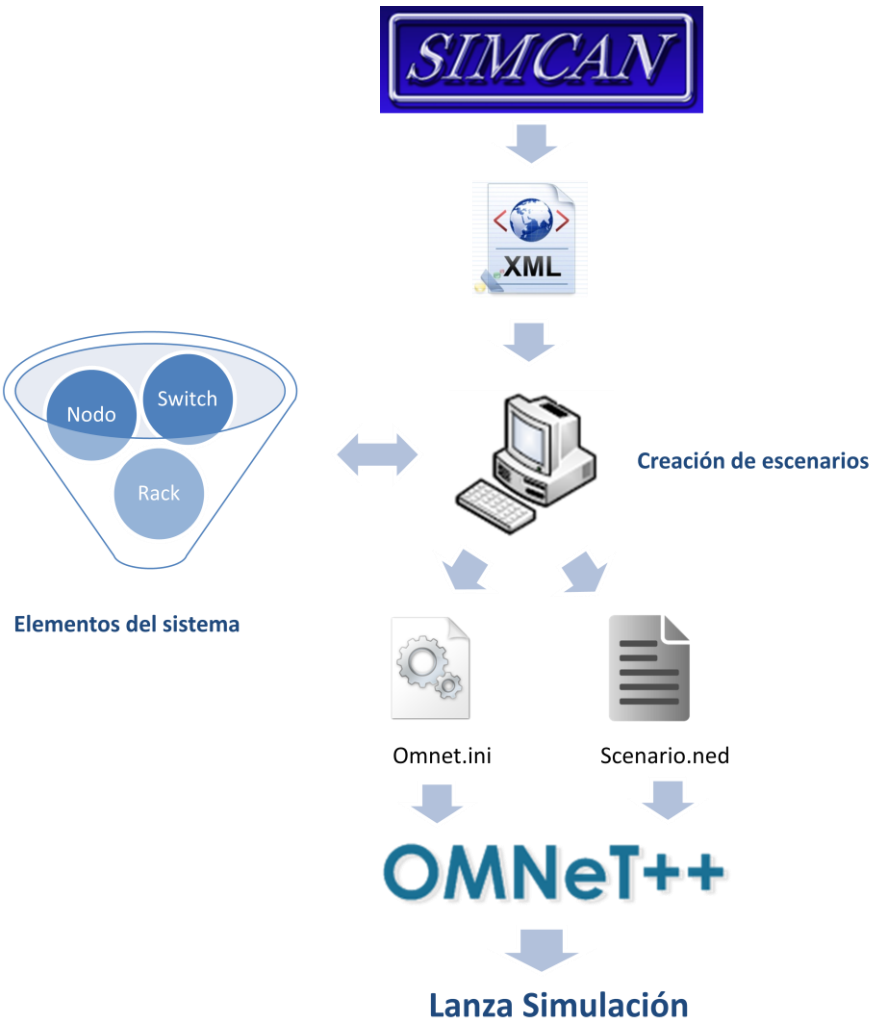

Ilustración 14 Flujo de información del sistema

5.4 Analizador sintáctico

Los archivos de la plataforma SIMCAN que contienen la información acerca de los parámetros de configuración son aquellos que poseen la extensión ".ned". La plataforma SIMCAN cuenta con un repositorio amplio y está estructurado por carpetas.

Para poder escanear los datos que se necesitan de la plataforma hay que recorrer todas las carpetas y subcarpetas. Por cada fichero encontrado que tenga la extensión previamente indicada, habrá que leer su contenido y obtener la información necesaria que es requerida por la aplicación.

Cada fichero con extensión ".ned" representa un módulo. Un módulo se define según las siguientes características:

- Tipo.
- Parámetros.
- Conexiones.
- Submódulos (sólo para módulos compuestos).

Según su tipo los módulos pueden ser simples o compuestos. Los parámetros configurables vienen definidos en el fichero por el nombre y el tipo de dato que contienen que puede ser numérico, cadena de caracteres o booleano. En la mayoría de los casos vienen descripciones sobre su uso en forma de comentario, que se almacenarán para posteriormente mostrárselas al usuario.

La siguiente acción existente para la configuración de un escenario son las conexiones. Pueden ser opcionales, en ellas se especificarán las entradas y salidas de cada módulo, como se conectan con otros elementos del escenario.

Los módulos compuestos se diferencian de los simples en que añaden, además de la información previamente descrita, submódulos. Los submódulos son referencias a otros módulos existentes en la plataforma y que contienen en su definición subparámetros. Los subparámetros contienen el valor que llevarán los parámetros del submódulo, el resto de parámetros podrán ser editadas por el usuario. Pueden ser valores literales o pueden venir expresados con el nombre de un parámetro del módulo principal, de este modo el valor del parámetro del submódulo adoptará el mismo valor que el que se defina en el

módulo principal. Los submódulos además contienen las conexiones de entrada y salida del propio submódulo, aunque al ser opcionales pueden no venir.

Toda esta información es necesario almacenarla en un fichero para su posterior uso, cuando los parámetros a configurar se muestren al usuario. Para poder manejar esta información, se almacenará en un fichero XML. De esta forma quedará estructurado, cada módulo junto con su tipo, parámetros, conexiones y submódulos.

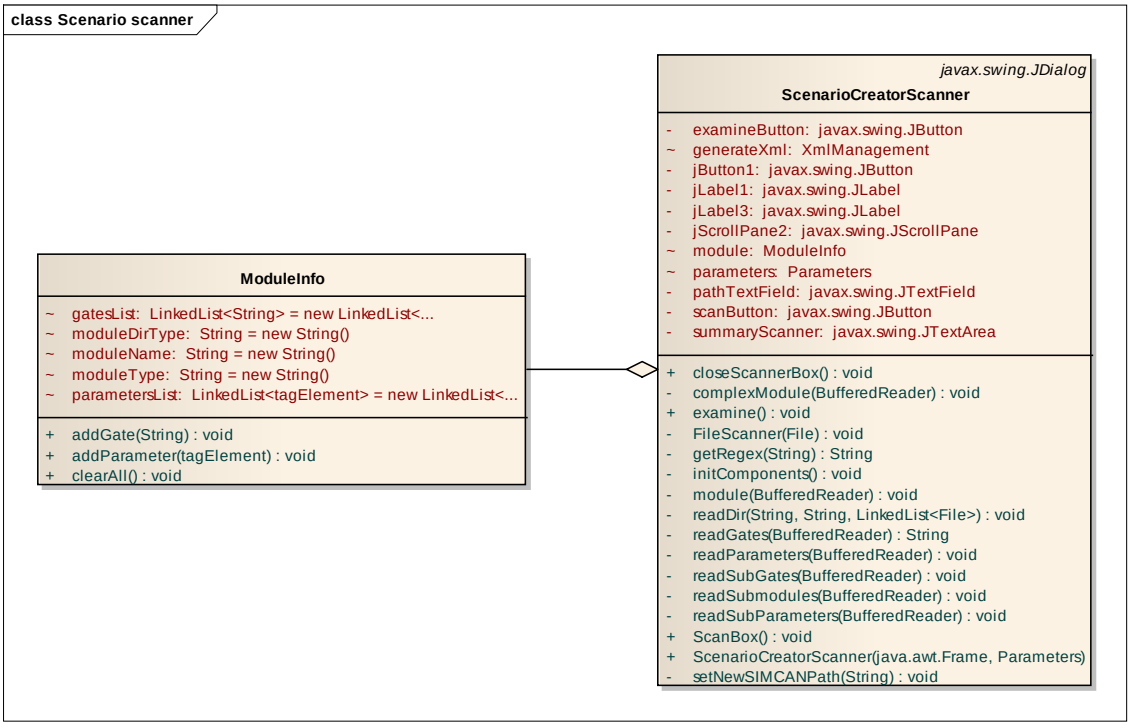

Ilustración 15 Modelo del scanner

El fichero XML generado al escanear la plataforma SIMCAN está sujeto a las variaciones o distintas versiones que se manejen de la propia plataforma. La opción de importar y exportar este fichero facilitará el control de versiones y su utilización como backups. Se podría volver a una versión previa siempre y cuando se exportara anteriormente.

Mediante estas opciones además se le ofrecerá al usuario la posibilidad de realizar una gestión más flexible de sus recursos. Además se posibilitará la

portabilidad entre diferentes equipos, y no será necesario tener una versión de la plataforma SIMCAN en cada equipo. De esta forma se podrá trabajar en diferentes entornos y distintos usuarios podrán intercambiar configuraciones personalizadas de manera sencilla.

5.5 Interfaz de usuario

El programa deberá disponer de una interfaz gráfica que facilitará al usuario interactuar con el sistema. Todos los parámetros y opciones se le presentarán al usuario mediante la interfaz gráfica, para guiarle a lo largo de la configuración del escenario y de los elementos que lo componen.

A continuación se describirán de forma pormenorizada las diferentes pantallas que formarán parte del sistema:

La pantalla principal se mostrará en el arranque de la aplicación y será la base sobre la que se configurarán los escenarios. Se compondrá básicamente de tres elementos:

- Barra de menú
- Árbol que contendrá todos los elementos almacenados en el sistema, clasificados por nodo, switch o rack.
- Layout donde se añadirán, organizarán e interrelacionaran los distintos elementos que componen el escenario.

La barra de menú contendrá todas las operaciones de aplicación expuestas en el capítulo anterior de análisis. Se distribuirán en tres grupos diferenciados además de la ayuda:

- Archivo

  - Salir de la aplicación.
  - Cargar fichero de repositorio.
  - Exportar fichero de repositorio.
- Herramientas
  - Escanear directorios.
- Escenarios
  - Nuevo escenario.
  - Cargar escenario.
  - Guardar escenario.
  - Guardar escenario como.
  - Generar archivos de configuración.
  - Habilitar simulación paralela.
  - Lanzar simulación.
- Ayuda
  - Información sobre la aplicación.

Los elementos que se agreguen al sistema aparecerán en el árbol de elementos dentro de su categoría correspondiente. Para ofrecer al usuario las máximas facilidades posibles en la gestión de este repositorio de elementos, se añadirán por medio de pop up menú (visible al hacer clic derecho con el ratón) las siguientes operaciones que fueron descritas en el apartado de análisis dentro de *Operaciones de Repositorio*:

- Crear nodo
- Editar nodo
- Eliminar nodo

- Añadir nodo al escenario
- Añadir n nodos al escenario

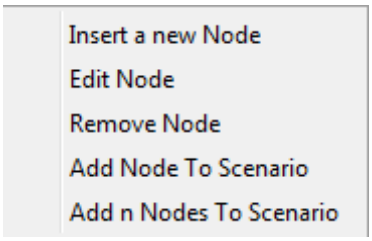

Ilustración 16 Pop up menú del repositorio

De forma análoga al elemento nodo, se muestran las operaciones correspondientes para los elementos switches y racks. Cada elemento tendrá su gestión diferenciada puesto que cada uno dispondrá de un tratamiento distinto en la aplicación.

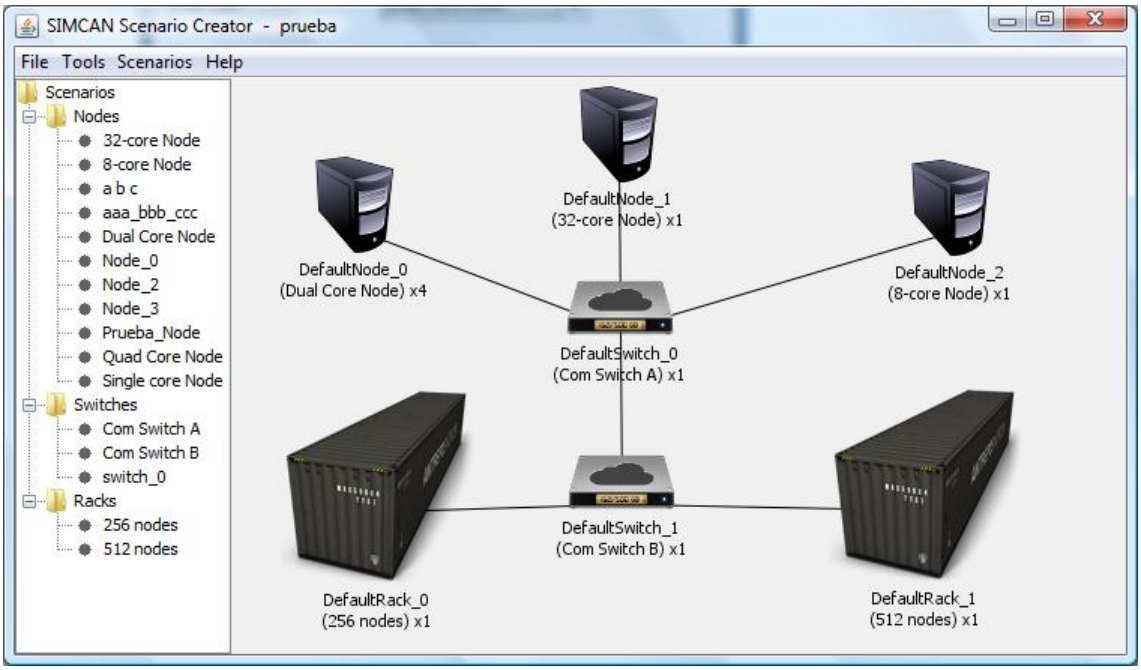

Ilustración 17 Pantalla principal

Como puede observarse en la ilustración anterior, el layout ocupará la mayor parte de la pantalla. En la parte izquierda se ubicará el árbol de elementos almacenados y en la parte superior el menú de la aplicación. Se persigue simplificar la interfaz de usuario para disponer del máximo espacio posible para crear escenarios. Además se pretende que la pantalla principal no resulte visualmente pesada al usuario, y a su vez que se disponga de todas las herramientas que ofrece la aplicación de una manera intuitiva.

Los elementos mostrados en la pantalla se compondrán de cuatro partes:

- Nombre en el escenario (Por defecto: Default + Tipo Elemento + Número).
- Tipo de nodo predefinido. El nombre se definirá entre paréntesis y se corresponderá con el elemento almacenado y mostrado en el árbol de elementos.
- Factor multiplicativo. Indica el número de elementos que contiene la representación. Se expresará precedido de una x, por ejemplo: x2.
- Imagen asociada al elemento.

Cada elemento dispondrá de una serie de operaciones asociadas para su representación en la interfaz. Dichas operaciones fueron expuestas en la sección de análisis, requisitos del sistema. Su uso se realizará por medio de un pop up menú, que al hacer clic con el botón derecho del ratón mostrará las siguientes operaciones:

- Renombrar elemento.
- Cambiar icono.
- Añadir conexión.
- Eliminar conexión.

- Eliminar elemento del escenario.

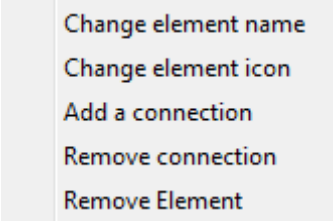

Ilustración 18 Pop up menú del escenario

Para renombrar un elemento se le mostrará al usuario un cuadro de diálogo (Ilustración 19) donde podrá escribir el nuevo nombre. Si el nombre introducido no está en uso se realizará el cambio y se mostrará actualizado en la interfaz. En caso contrario se mostrará un error por pantalla, señalando que el nombre ya está siendo utilizado en el escenario actual.

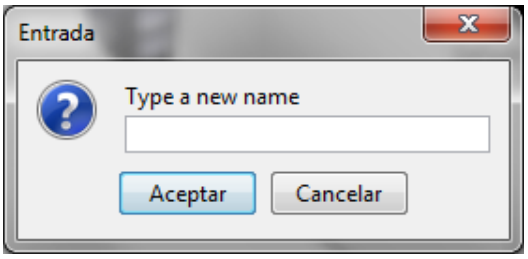

Ilustración 19 Renombrar elemento

Para poder modificar la imagen asociada a un elemento, se deberá seleccionar y elegir la imagen a cambiar, que puede encontrarse en cualquier carpeta del sistema. Para facilitar la elección se previsualizarán las imágenes seleccionadas antes de aceptar la modificación.

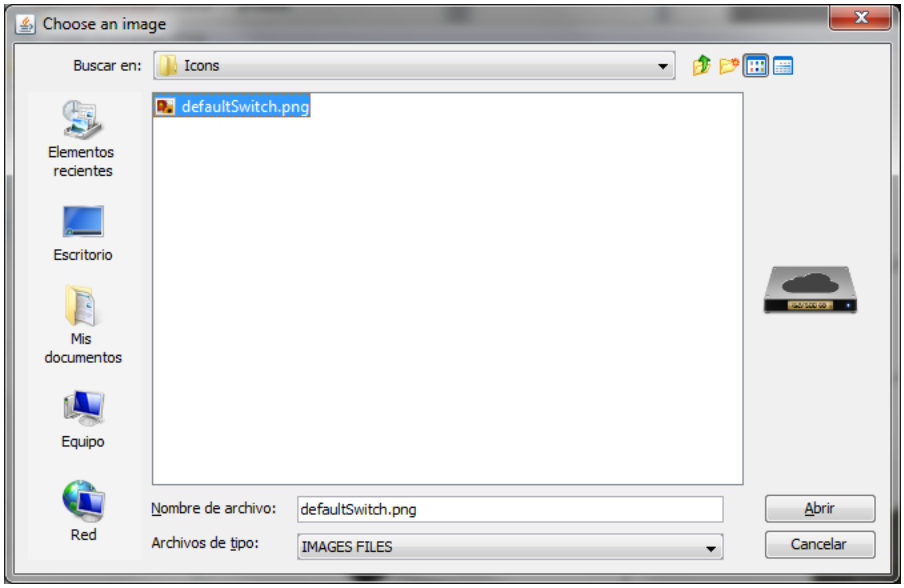

Ilustración 20 Cambiar imagen del elemento

Para establecer conexiones entre los elementos se ha de crear una pantalla (Ilustración 21) que permita al usuario seleccionar el elemento destino (se debe seleccionar previamente un elemento origen para tener acceso al pop menú del escenario) además del tiempo de retardo y la tasa de transferencia.

Con el objetivo de guiar las conexiones entre elementos, la aplicación ha de restringir las conexiones que no estén permitidas. La forma de realizar las conexiones será por medio de la selección de uno de los dos elementos que se quieran conectar. Una vez seleccionado un elemento se mostrará un listado del resto de elementos susceptibles de ser conectados.

En caso de no existir ningún elemento en el escenario que cumpla las condiciones, el listado aparecerá vacío y no se podrá establecer ninguna conexión. Si un elemento rack o nodo ya tienen establecida una conexión con un elemento switch se inhabilitarán para poder conectarse a otro switch, puesto que el número máximo de conexiones de estos elementos será de uno.

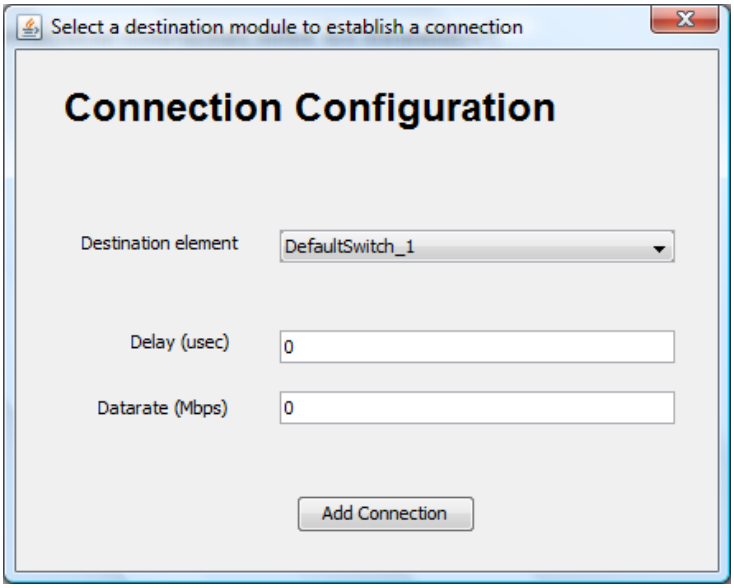

Ilustración 21 Configurador de conexión

En el caso de querer eliminar una conexión establecida hay que tener en cuenta que debido al hecho de que un elemento swtich puede tener múltiples conexiones se creará un gestor para habilitar esta funcionalidad (Ilustración 22).

El usuario seleccionará un elemento de los dos que forman la conexión y seleccionará la opción de eliminar una conexión. Se mostrarán al usuario todas las conexiones establecidas que tiene dicho elemento, identificando las conexiones por medio del elemento destino, que siempre será el otro elemento que forme parte de la conexión.

En el supuesto de que no existiesen conexiones, el listado saldrá vacío y no se permitirá realizar ninguna acción.

Con el fin de evitar eliminaciones múltiples de conexiones por error, sólo se podrá eliminar una única conexión por acción. Para eliminar distintas conexiones de un elemento habrá que repetir el proceso y seleccionar la conexión a eliminar.

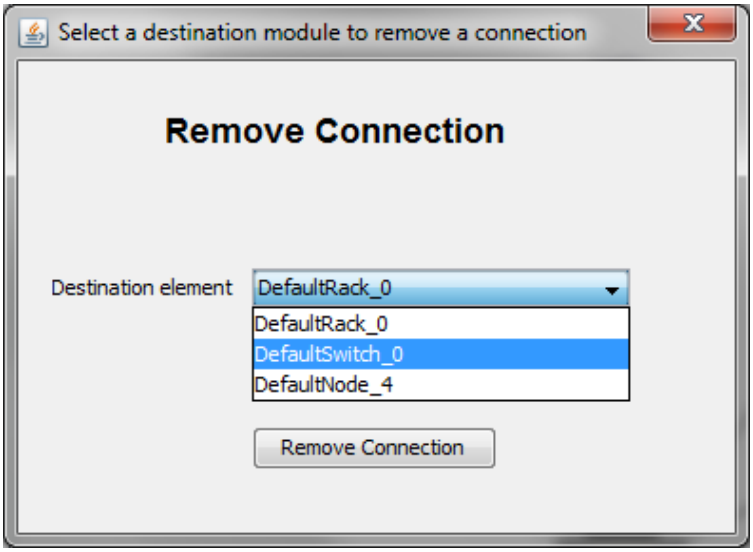

Ilustración 22 Eliminar una conexión

Para escanear el directorio donde se encuentra la plataforma de comunicación SIMCAN y de esta forma obtener el fichero de configuración, se creará una pantalla desde donde el usuario podrá llevarlo a cabo. Se mostrará un cuadro de texto donde introducir la ruta del directorio de SIMCAN o se podrá obtener mediante el explorador de directorios que el sistema operativo tenga instalado.

Además esta pantalla mostrará un log donde se exponga la información referente a los módulos que se han añadido al fichero de configuración. En caso que se produzca un error también será recogido por el log para informar al usuario.

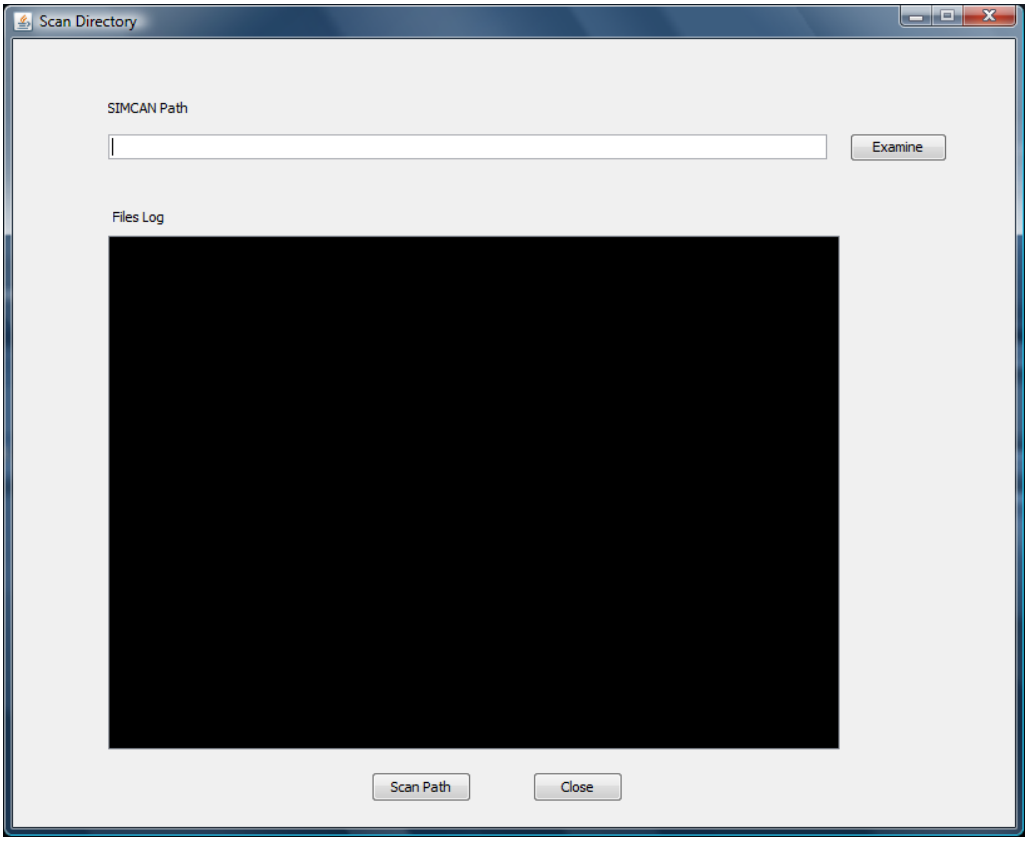

Ilustración 23 Escaneo de directorios

Configuración de un nodo.

Las funciones de crear y editar nodo compartirán las pantallas de configuración aunque el tratamiento interno será sensiblemente distinto. La configuración de un nodo se dividirá en cuatro partes diferenciadas:

- Sistema operativo.
- CPUs.
- Block servers.
- Aplicaciones.

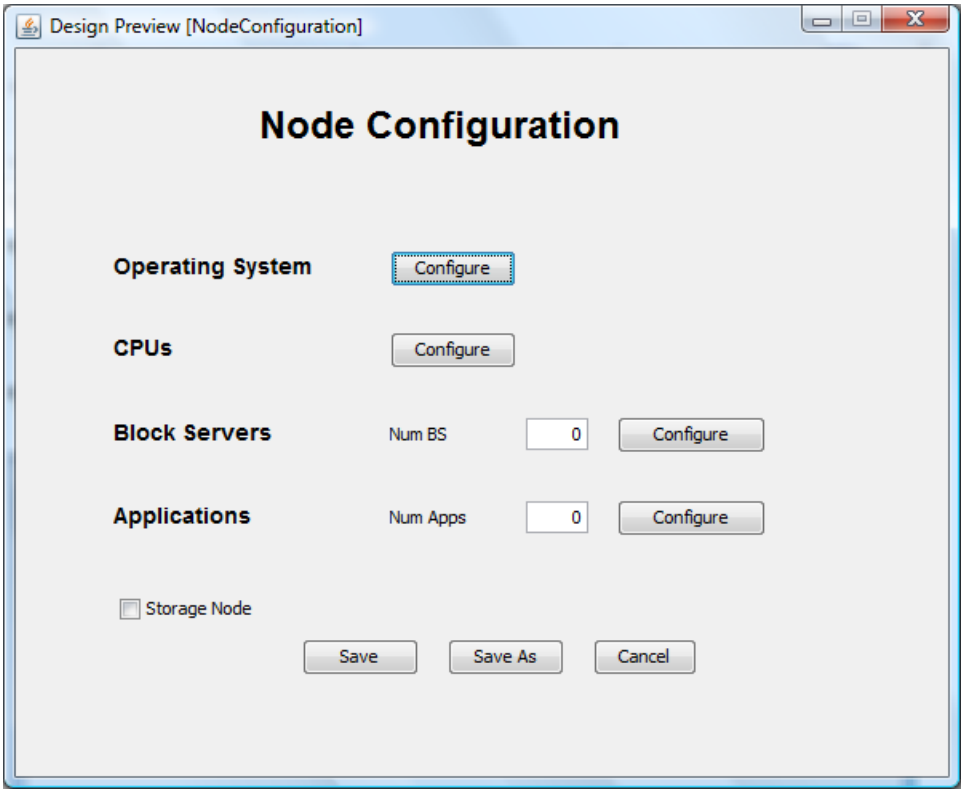

Ilustración 24 Configuración del nodo

Además mediante una casilla de verificación el usuario podrá especificar si se trata o no de un nodo de almacenamiento. Para completar la edición o creación del nodo, todas las partes que lo forman habrán de contener datos válidos, sino se mostrará el error mediante un cuadro de diálogo. El usuario podrá comprobar que todos los elementos son correctos si todos los botones de configuración están en verde. Si alguno no ha sido editado aún, no presentará ningún color y si los datos son incorrectos se mostrará en rojo.

Sistema operativo.

En la configuración del sistema operativo se mostrará un cuadro de diálogo con los parámetros a configurar obtenidos de la plataforma de simulación SIMCAN.

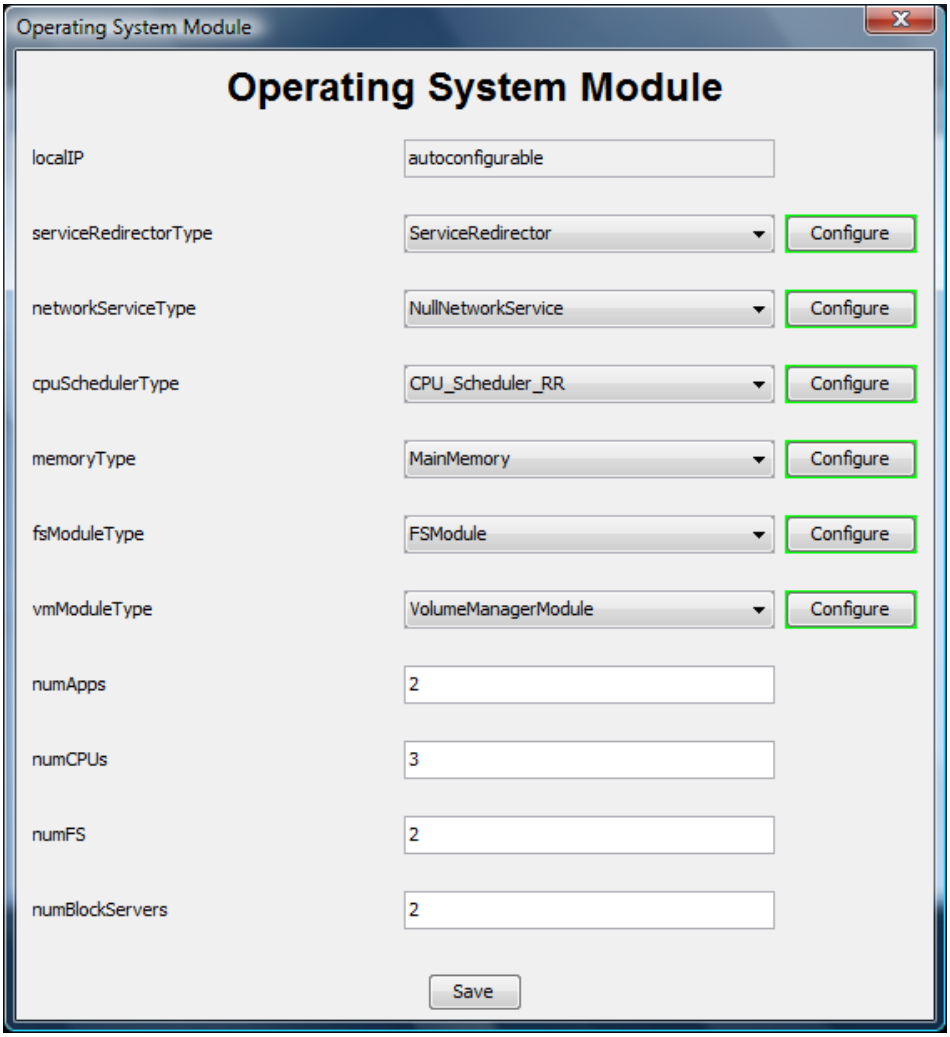

Ilustración 25 Configuración de Sistema Operativo.

La interfaz del sistema operativo se generará dinámicamente en función de los parámetros contenidos en la plataforma de simulación SIMCAN. Los parámetros se mostrarán al usuario de manera distinta según el tipo, así se facilitará al usuario la configuración de los mismos. Para los parámetros tipo, que se configuran por medio del submódulo, aparecerá en la parte izquierda el nombre (al poner el cursor por encima se mostrará la descripción del parámetro según está en SIMCAN). En la parte central aparecerá una pestaña con los diferentes tipos de submódulos definidos en SIMCAN que corresponden al tipo del parámetro. En la parte derecha se creará un botón con la opción de configurar una vez elegido el tipo de submódulo. Una vez presionado

aparecerá un cuadro de diálogo con los parámetros a configurar del submódulo elegido, si los datos son correctos al guardar los cambios se pondrá a verde el botón de configuración. Sino seguirá sin color y por tanto pendiente de configurar.

El otro tipo de parámetro que forma parte del sistema operativo es el simple. La parte izquierda es exactamente igual que en el caso anterior, pero en la parte central aparecerá un cuadro de texto para introducir el valor de dicho parámetro, en vez de un combobox con las opciones predefinidas en SIMCAN. No se permitirá que se introduzcan tipos no válidos como una letra en un tipo numérico, y se notificará por cuadro de diálogo al usuario el error cometido cuando trate de guardar los cambios del sistema operativo.

Existe dos excepciones en la configuración de los parámetros tipo, una corresponde al sistema de archivos, debido a que consta de listados de parámetros a completar por el usuario. Quedará dispuesto según la ilustración siguiente para simplificar y hacer más sencillo la configuración del sistema de archivos.

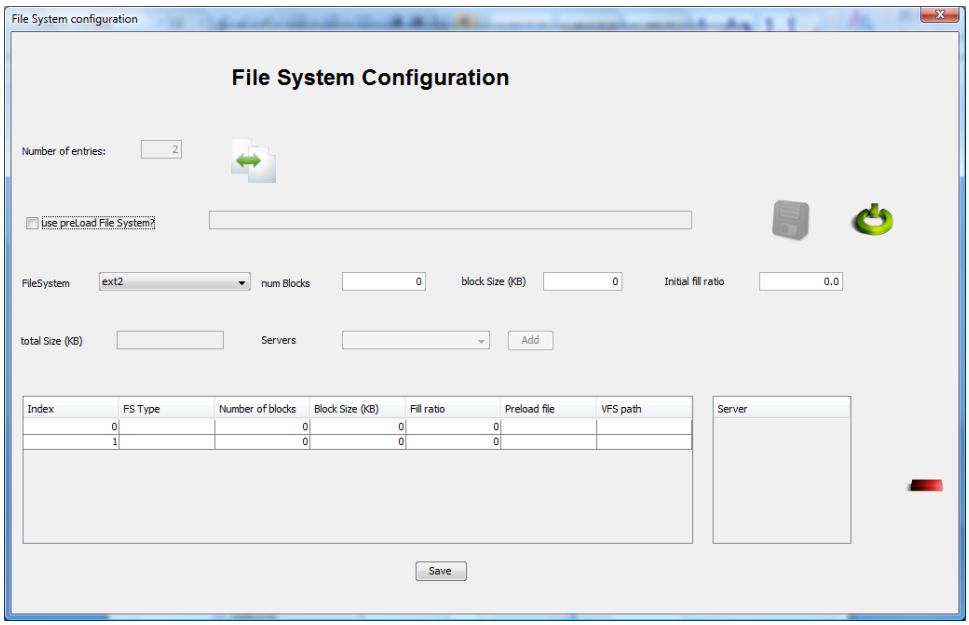

Ilustración 26 Configuración del sistema de archivos

La pantalla de configuración del sistema de archivos constará de un número de entradas que definirán el número de filas de de la matriz de parámetros. El número de entradas no será editable en esta pantalla por el usuario, debido a que el número de sistemas de archivos se definirá en la pantalla anterior, correspondiente a la configuración del sistema operativo.

El usuario tendrá la opción de configurar el sistema de archivos dependiendo del tipo que elija. Para ello se mostrará un listado con los sistemas de archivos declarados en SIMCAN, el usuario sólo deberá seleccionar el que más se adecue a la configuración que desee llevar a cabo. También deberá rellenar el número de bloques, el tamaño de cada bloque y el initial fill ratio.

El usuario también dispondrá de la posibilidad de pre cargar un fichero del sistema de archivos o generar uno.

Finalmente el usuario podrá seleccionar el nodo de almacenamiento en el campo server. Sólo se activará esta opción para el tipo de sistema de ficheros NFS o Parallel, y el listado seleccionable se rellenará con los nodos de almacenamiento dispuestos en el escenario actual.

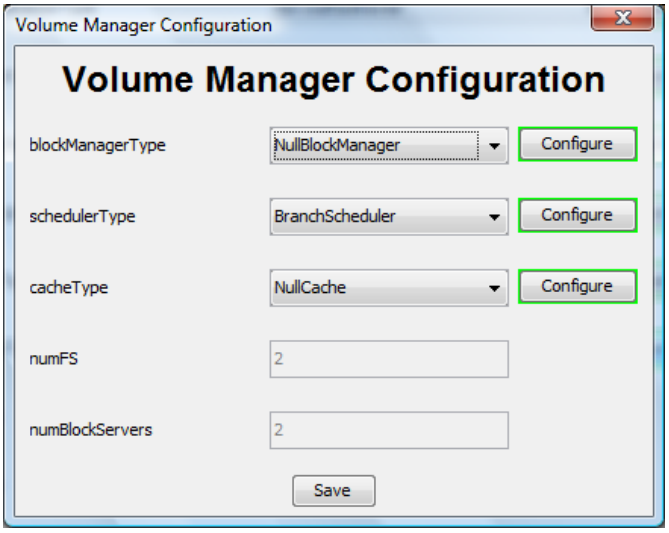

Ilustración 27 Administrador de volúmenes

La otra peculiaridad que se da en la configuración del sistema operativo se produce con el administrador de volúmenes.

A diferencia del resto de parámetros configurables del sistema operativo, el administrador de volúmenes contiene a su vez parámetros que se configuran a partir de submódulos, que contienen sus propios parámetros. Con el objetivo de mantener la misma estructura y orden lógico que en el resto de elementos, se creará una pantalla intermedia que será la de administrador de volúmenes.

Esta pantalla se generará dinámicamente, de manera análoga a como se genera la pantalla del sistema operativo. Leerá la información del fichero obtenido de SIMCAN, para generar los parámetros configurables. Como se puede apreciar en la ilustración anterior, albergará parámetros configurables a partir de otros submódulos y parámetros simples.

CPUs

La configuración de la CPU se realizará de una manera directa, a diferencia del sistema operativo que como se explicó anteriormente, tenía un tratamiento especial en algunos puntos.

Para este caso se obtendrán los parámetros de la plataforma de simulación SIMCAN y se generará de manera dinámica una ventana con la el nombre del módulo a configurar, en este caso CPU, y se mostrarán los parámetros obtenidos de SIMCAN. Para los parámetros tipo se procederá como se ha explicado en los casos anteriores: se mostrará un listado de los elementos disponibles correspondientes al tipo. El usuario seleccionará el que se ajuste a sus necesidades y al configurar se mostrará una ventana con los parámetros a rellenar de dicho elemento.

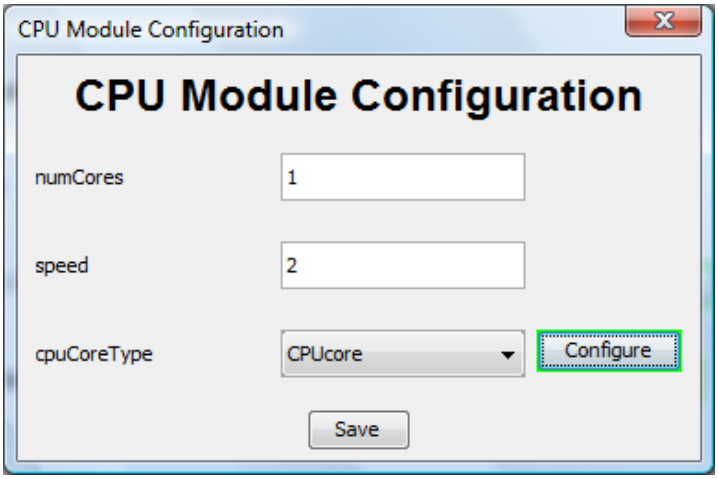

Ilustración 28 Configuración del módulo CPU

Block servers

La configuración del block server es exactamente igual al explicado anteriormente en el caso de la CPU. Se obtienen los parámetros de la plataforma de simulación SIMCAN y se genera de manera dinámica un formulario con los datos a rellenar por parte del usuario.

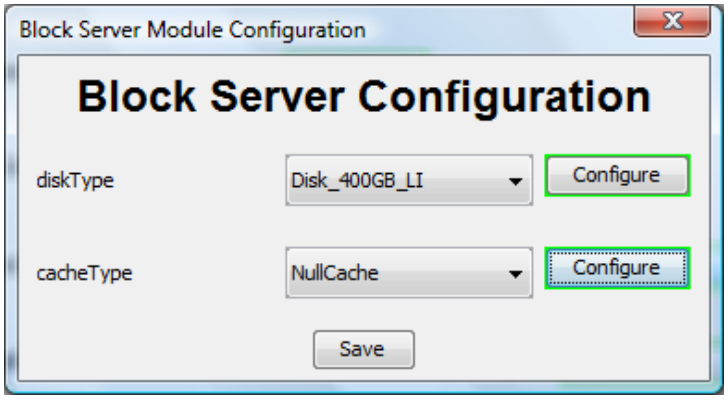

Ilustración 29 Configuración del módulo Block Server

Aplicaciones

Las aplicaciones se generarán de una forma ligeramente diferente al resto de módulos que son necesarios para configurar el nodo. Requiere que previamente

(en la pantalla de configuración del nodo) se especifique el número de aplicaciones que se configurarán. Se generará de forma dinámica una ventana con tantas aplicaciones como especifique el usuario.

Cada aplicación será tratada como un parámetro tipo, donde el usuario tendrá la opción de elegir entre las distintas aplicaciones definidas en la plataforma de simulación SIMCAN.

Se configurará cada aplicación de manera independiente, y una vez elegido el tipo de aplicación se podrán rellenar los parámetros correspondientes a dicha aplicación.

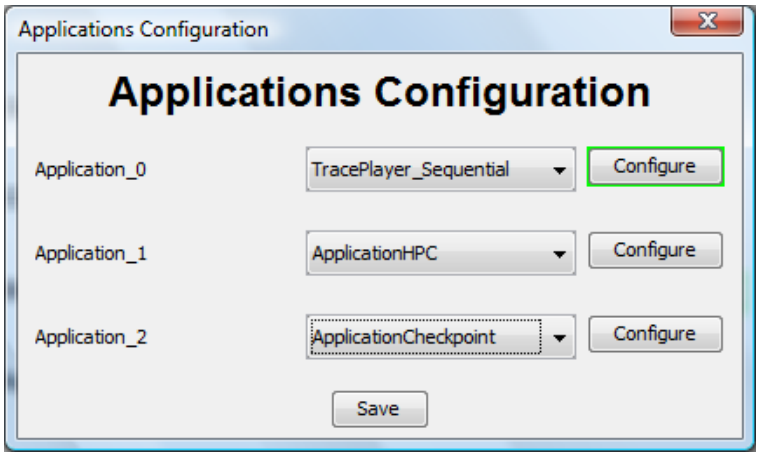

Ilustración 30 Configuración del módulo de Aplicaciones

Configuración de un switch.

La configuración de un elemento switch es mucho más sencilla que la de un nodo, expuesta anteriormente. En este caso no se dividirá en partes, sino que se mostrará de manera estática una serie de parámetros a rellenar por parte del usuario como se expone en la ilustración siguiente.

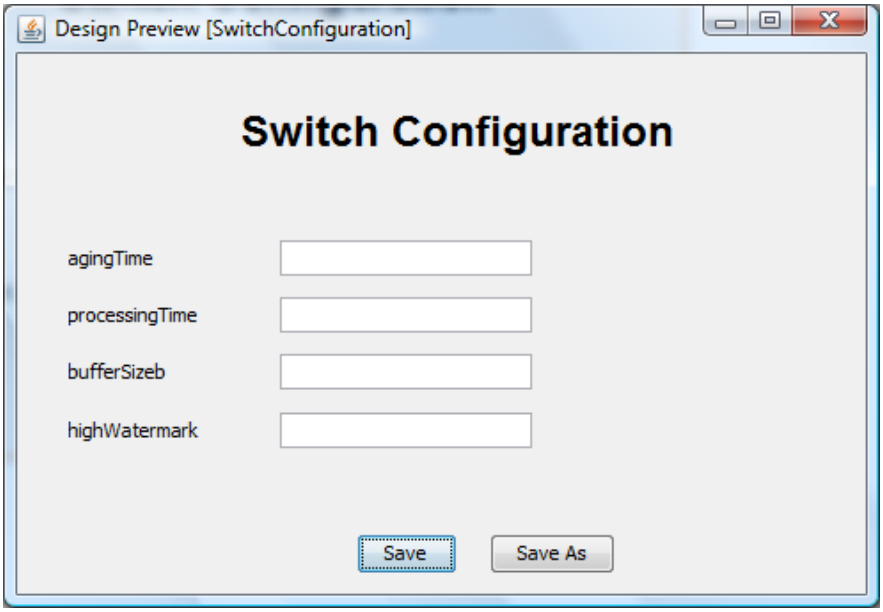

Ilustración 31 Configuración de un Switch

Configuración de un rack.

La configuración del Rack se realizará de forma dinámica, los parámetros se obtendrán de la plataforma de simulación SIMCAN. La ventana que se mostrará al usuario se creará en función de los parámetros leídos. En el caso de existir un parámetro tipo se procederá como en los casos anteriores: se mostrará un listado con los elementos que se corresponden con el tipo. Una vez el usuario realice la selección del tipo procederá a su configuración presionando el botón de configuración que mostrará otra ventana con los parámetros a listos para establecer su valor.

Un rack está formado por un determinado número de nodos (del mismo tipo), por esta razón se mostrará al usuario un listado de los nodos a elegir. Esta lista de nodos estará formada por los nodos que el usuario haya definido previamente, como se expuso con antelación en la configuración de un nodo.

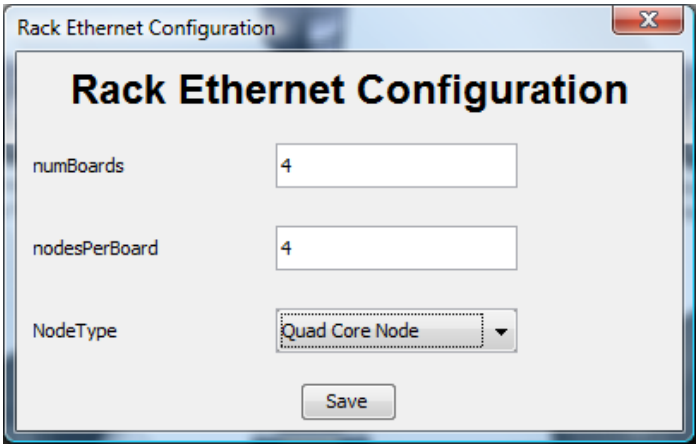

Ilustración 32 Configuración de un Rack

5.6 Sintaxis del fichero de configuración del sistema

En este apartado se va a explicar cómo está definido el fichero de configuración del sistema que se genera a partir de la información contenida en la plataforma de simulación SIMCAN. Este fichero contendrá toda la información relevante para la aplicación de la plataforma SIMCAN para poder tratar los datos de una forma más eficiente.

Aunque existen diversas sintaxis posibles que nos permitirían almacenar la información en disco, muchas de ellas dependen directamente del lenguaje de programación utilizado en la implementación de la propia herramienta. Debido a ello, se ha decidido utilizar un estándar para el intercambio de información estructurada como es el lenguaje XML (eXtensible Markup Language), ya explicado más detalladamente en el apartado 2.7 del presente documento.

Como ya se expuso anteriormente, la información que contiene la plataforma de simulación SIMCAN, repartida en distintos directorios, es necesario obtenerla y almacenarla en un formato estructurado que permita realizar consultas sobre

los distintos módulos y parámetros que lo forman. Los ficheros que contienen dicha información son los que tienen la extensión ".ned".

Debido a que el proceso de configuración, que guía al usuario a parametrizar los distintos elementos que forman un escenario, realiza múltiples consultas a dicha información. Se deberá almacenar los datos estrictamente necesarios y a su vez que se puedan consultar de una manera rápida y sencilla.

Por estas razones se ha elegido estructurar el fichero de forma jerárquica, para buscar los elementos necesarios en cada momento y no tener que recorrer el fichero cada vez de inicio a fin.

En la Ilustración 33 se puede observar un fichero del repositorio de SIMCAN con extensión ".ned". De todos estos ficheros se extraerá la información para crear el fichero de configuración del sistema.

```
import   "GenericCPUcore";
module CPU_Module
 parameters:
        numCores: numeric const,        // Number of CPU cores
        speed: numeric const,           // CPU core speed (in MIPS)
        cpuCoreType: string;            // Type of each CPU core
  gates:
        in: fromOS[];                   // Input gates from Operating System
        out: toOS[];                    // Output gates to Operating System
  submodules:
        CPUcore: cpuCoreType [numCores] like GenericCPUcore;
                parameters:
                        speed = speed;
  connections nocheck:

        // Connections between all CPU cores and outside
        for i=0..numCores-1 do
                CPUcore[i].in <-- fromOS[i];
                CPUcore[i].out --> toOS[i];
        endfor;
endmodule
```

Ilustración 33 Fichero CPU_Module.ned

En la Ilustración 42 se puede apreciar un ejemplo del fichero de configuración. Se han mostrado dos módulos, un simple y otro complejo para ilustrar el formato. El módulo complejo que hace referencia al sistema operativo, se ha recortado parte del contenido con el ánimo de reducir su volumen sin que el formato se vea perjudicado.

La estructura de un módulo simple será la siguiente:

- Module.
    - Name
    - Type
    - Paremeter
    - Gate

La etiqueta module es la etiqueta raíz de cada módulo. Contiene toda la información relevante distribuida entre sus etiquetas hijas. Dicha etiqueta posee el atributo *Dir_Type* que especifica la carpeta del repositorio de SIMCAN donde se encuentra el fichero que contiene la definición del módulo. La información contenida en dicho atributo será muy útil puesto que permitirá buscar los elementos que pertenecen a un tipo determinado.

Por ejemplo en la Ilustración 42 se muestra parte de la información de SIMCAN acerca del módulo del sistema operativo. En la configuración del parámetro *serviceRedirectorType*, se recuperarán (del fichero de configuración de SIMCAN) todos los módulos cuyo atributo Dir_Type corresponda a "ServiceRedirectors". De esta forma se mostrará al usuario un listado con todos los elementos existentes en SIMCAN que se pueden utilizar en la configuración de dicho parámetro. El usuario sólo deberá elegir uno y completar los datos requeridos por el sistema.

La etiqueta *Name* contendrá el nombre del módulo en cuestión, y la etiqueta *Type* especificará si se trata de un módulo simple (simple) o compuesto (module).

La etiqueta *Parameter* definirá la información relativa a cada parámetro y aparecerá tantas veces como parámetros tenga el módulo. Contendrá los siguientes atributos:

- Name: Nombre del parámetro.
- Type: Tipo de dato en el que se expresará el parámetro (numérico, alfanumérico o booleano).
- Comment: Breve definición acerca del parámetro. Será útil almacenarlo para poder mostrárselo al usuario en la interfaz de configuración.

La etiqueta *Gate* hará referencia a las conexiones que se podrán definir para cada módulo. Contendrá dos atributos:

- Name: Nombre de la conexión
- Type: Tipo de conexión, de entrada o de salida. (in, out)

Una vez definida la estructura un módulo simple se hará lo mismo para un módulo compuesto. El módulo compuesto requiere la misma estructura que el módulo simple pero además añade más etiquetas correspondientes a los submódulos que contiene, quedaría de la siguiente manera:

- Module.
    - Name
    - Type
    - Paremeter
    - Gate

- Submodule
    - Subname
    - Subtype
    - Subparameter
    - Gatesize

La etiqueta *Submodule* se comportará de manera similar a la etiqueta *Module*, engloba las etiquetas que contienen los datos relativos al submódulo correspondiente y se definen a continuación:

Subname es la etiqueta que contendrá el nombre del submódulo al que se hace referencia.

Subtype es la etiqueta que especificará en el módulo en que se basa el referido submódulo.

Subparameter es la etiqueta que definirá el valor que tomará cada parámetro del submódulo por defecto. Posee los siguientes atributos:

- Name: Nombre del subparámetro.
- Valor: Valor que tomará el subparámetro.

Los valores que puede tomar son una constante (un valor fijo) o una variable, como un parámetro del módulo al que pertenece. Esto significa que el usuario no podrá editar este valor directamente, dado que este parámetro se configurará de manera automática por el sistema. El resto de parámetros que contenga el submódulo pero que no está definidos como subparámetros si serán configurables directamente por el usuario.

Gatesize es la etiqueta que hará referencia a las conexiones que se podrán definir para cada submódulo.

5.7 Sintaxis de los ficheros de almacenamiento de los elementos del sistema

A continuación se procede a describir el formato de los ficheros de almacenamiento de los elementos. La información será introducida por el usuario y que generará la herramienta a desarrollar. Estos ficheros de almacenamiento son aquellos en los que se van a recoger los datos asociados a los componentes creados en el sistema, es decir: los nodos, los switches, los racks y los escenarios.

Los ficheros de almacenamiento, al igual que el fichero de configuración, se guardarán acorde al formato **XML.** La información de los elementos que se van a guardar se ha de estructurar de forma jerarquizada. Es debido a que se pueden modificar algunos aspectos de los contenidos y se realizarán consultas, como en la obtención de los datos para la generación de los archivos de simulación.

Sin embargo para almacenar los escenarios se hará mediante serialización. Un escenario, en el contexto de esta herramienta, es una representación de todos los elementos y sus interconexiones. Para almacenar todos los componentes, coordenadas y conexiones es la forma más eficiente de llevarlo a cabo.

Como se narró en apartados anteriores, la herramienta va a trabajar con varios elementos fundamentales, cuya información deberá almacenar y recuperar fácilmente: los nodos, los switches, los racks y los escenarios. Cada uno de estos elementos se gestionará por separado en la herramienta, por esta razón su almacenamiento también será independiente. Así, nos encontramos con la existencia de un conjunto de cuatro tipos distintos de archivos de almacenamiento.

Además, para facilitar el acceso y recuperación de esta información, se ha decidido gestionar estos archivos en distintos directorios, de acuerdo al tipo de archivo que contenga cada uno de ellos.

```
<Node>
 <APPS>...</APPS>
 <BS>...</BS>
 <CPU>...</CPU>
 <OS>...</OS>
 <Storage Value="false"/>

</Node>
```

Ilustración 34 Fichero de almacenamiento de un Nodo

En la Ilustración 34 se puede observar la estructura en la que se basa la configuración de un nodo, se divide de la siguiente manera:

- Aplicaciones: Etiqueta *APPS*
- Block Server: Etiqueta *BS*
- CPU: Etiqueta *CPU*
- Sistema operativo: Etiqueta *OS*
- Si es un nodo de almacenamiento: Etiqueta *Storage*, Atributo Value (valores posibles true, false).

Para una mejor comprensión se detallarán por separados los componentes que forman un nodo.

Las aplicaciones se definirán mediante la etiqueta *Application_x*, donde la x será un valor numérico para diferenciar las distintas aplicaciones. Contendrá un atributo *Type* que especificará a que aplicación definida en la plataforma SIMCAN hace referencia. Se puede apreciar en la Ilustración 35.

```
<APPS>
 <Application_0 Type="LocalApplication">
  <Parameter Name="application_netType" Type="string" Value="INET"/>
  <Parameter Name="startDelay" Type="numeric const" Value="2"/>
 </Application_0>
 <Application_1 Type="LocalApplication">
  <Parameter Name="application_netType" Type="string" Value="INET"/>
  <Parameter Name="startDelay" Type="numeric const" Value="10"/>
 </Application_1>
 <APPSParameter Name="numAPPS" Value="2"/>
</APPS>
```

Ilustración 35 Detalle de aplicaciones de un nodo

Cada aplicación contendrá los parámetros correspondientes mediante la etiqueta *Parameter*, que habrá tantas como parámetros contenga. Tendrá los siguientes atributos:

- Name: Nombre del parámetro.
- Type: Tipo de dato en el que se expresará el parámetro (numérico, alfanumérico o booleano).
- Value: Valor que adoptará el parámetro.

Finalmente la etiqueta APPSParameter detallará el parámetro global que especifica el número de aplicaciones que se definirán. Tendrá los siguientes atributos:

- Name: Nombre del parámetro.
- Value: Valor que adoptará el parámetro.

A continuación se detallará la información referente al block server de un nodo, como se recoge en la Ilustración 36.

La etiqueta BSParameter contendrá los parámetros sencillos que no se completan a partir de otro módulo (parámetros tipo), sino por los siguientes atributos:

- Name: Nombre del parámetro.
- Type: Tipo de dato en el que se expresará el parámetro (numérico, alfanumérico o booleano).
- Value: Valor que adoptará el parámetro.

```
<BS>
 <BSParameter Name="numBS" Type="numeric" Value="2"/>
 <diskType Type="Disk_400GB_LI"/>
 <cacheType Type="NullCache">
  <Parameter Name="numInputs" Type="numeric const" Value="1"/>
 </cacheType>
</BS>
```

Ilustración 36 Detalle de block server de un nodo

Después se escribirán las etiquetas que definen a los parámetros tipo (se diferencian porque el nombre del parámetro siempre acaba en Type). El nombre de la etiqueta será el parámetro y el atributo que las define es Type, que hará referencia al elemento de la plataforma SIMCAN que será seleccionado por el usuario.

A un nivel inferior, mediante la etiqueta *Parameter*, se escribirán todos los parámetros que definen al elemento seleccionado. Los atributos son los mismos que en el resto de los casos que se definen parámetros, name, type y value, previamente detallados.

La forma en que se almacena la información referente a la CPU de un nodo es igual que la forma de guardar el block server. La única diferencia radica en el nombre de la etiqueta que guarda los parámetros simples, en este caso *CPUParameter*.

Se realiza esa diferenciación para no confundir los parámetros de primer orden con los pertenecientes a un segundo orden que si se escriben con la etiqueta

Parameter. De esta manera la forma de consultar o editar datos es más sencilla y clara. Se puede observar lo expuesto en la Ilustración 37.

```
<CPU>
 <cpuCoreType Type="CPUcore">
  <Parameter Name="speed" Type="numeric const" Value="5"/>
  <Parameter Name="tick_s" Type="numeric const" Value="6"/>
 </cpuCoreType>
 <CPUParameter Name="numCores" Type="numeric const" Value="1"/>
 <CPUParameter Name="speed" Type="numeric const" Value="2"/>
</CPU>
```

Ilustración 37 Detalle de CPU de un nodo

El sistema operativo será el elemento más complejo de configurar y almacenar de todos los que componen el nodo. Se debe a que contiene un número elevado de parámetro y hasta tres niveles de profundidad.

Para mantener la homogeneidad y seguir la misma norma que en los elementos anteriormente descritos, la etiqueta que contiene los parámetros sencillos de primer orden del sistema operativo se llamará *OSparameter*. Al igual que el resto de etiquetas de parámetros estará compuesta por los atributos ya descritos anteriormente Name, Type y Value.

Los parámetros tipo (se configuran a partir de otro módulo) se escribirán, al igual que en los casos anteriores, con nombre de etiqueta igual a su propio nombre. Tendrán como atributo Type, que detallará el módulo o elemento en el que se basan y habrá sido seleccionado por el usuario. Los parámetros que contengan serán definidos por la etiqueta *Parameter*, que tendrán los atributos ya descritos para el caso de los parámetros.

En la Ilustración 43 se puede apreciar que en el Volumen manager (vmModuleType) se alcanza el nivel tres de profundidad de parámetros. Es debido a que contiene parámetros tipo que a su vez contienen parámetros. Estos

parámetros se escribirán con la etiqueta *Subparameter* para diferenciarlos del resto y hacer más sencillas las consultas o modificaciones.

El único módulo que se configurará de manera distinta, como se explicó en el análisis y en la parte de interfaz del diseño, es el file system. Los datos se grabarán como una tabla de elementos, y cada fila se escribirá mediante la etiqueta FSROW, que contendrá los siguientes atributos:

- Blocksize
- FSType
- FillRatio
- Index
- NumBlocks
- PreloadFile
- VFSPath

5.8 Sintaxis de los ficheros de salida

Se denominarán ficheros de salida a aquellos ficheros que se generarán con la sintaxis requerida por el simulador OMNeT++. Estos ficheros contendrán toda la información relativa a un escenario, tanto los elementos que lo forman como las conexiones existentes.

Por cada escenario se escribirán lo siguientes ficheros de salida:

- Omnetpp.ini
- Scenario.ned

5.8.1 Omnet.ini

En el fichero omnetpp.ini se escribirán todos los elementos, donde se detallarán todos sus parámetros de configuración. La sección en la cual se describen la parte general constará de los puntos siguientes y son necesarios para la simulación en OMNET:

- Las rutas donde se encuentran los archivos preload-ned-files.
- Si se habilita o la simulación en paralelo.
- Si se habilitan los vectores de salida (output vectors)
- Si se habilita la opción modo express (sin mensajes).
- La ruta donde se encuentran los plugins.

En la siguiente versión del fichero se describirán los elementos que forman parte del escenario. Se definirán los parámetros que forman cada nodo, switch y rack. La descripción se llevará a cabo siguiendo la notación requerida por OMNET, para que el usuario se despreocupe de editar el fichero.

El fichero también contendrá el rango que se le asignará a cada aplicación contenida en cada node de cada nodeboard de los distintos racks que forman el escenario (en caso que los hubiere). El valor será incremental, se partirá de 0 y cada uno recibirá un valor distinto.

Finalmente se asignarán las direcciones MAC, que se escribirán de la siguiente manera:

- Una por cada nodo de cada nodeboard que contiene cada rack.
- Una por cada nodo de almacenamiento.
- Una por cada nodo simple.
- Una por cada switch en el rack

- Una por cada conexión del swtich.

Para poder explicar de forma pormenorizada las distintas secciones que forman los ficheros, se abordará el seguimiento de los ficheros por partes.

La cabecera del fichero contendrá los datos generales, como el nombre del escenario, las rutas de los ficheros preload, si se trata de una simulación paralela, etc.

En la Ilustración 45 se puede ver cómo se definen los nodos de un escenario. La estructura se basa en el nombre del escenario, el nodo tratado y los distintos niveles de módulos que lo forman hasta llegar al valor final de los parámetros.

Se escribirán los parámetros previamente configurados por el usuario. Para facilitar su comprensión se añaden comentarios para identificar los módulos escritos.

Los racks que forman parte del escenario se escribirán de forma análoga a la descrita en los nodos. Esto es, todos los parámetros que lo componen, entre ellos la definición completa del nodo de la que está compuesto el rack

La definición de cada switch se realizará de la misma manera que para un nodo o para un rack. Se leerán todos los parámetros almacenados en el fichero de configuración del switch y se escribirán, según el formato demandado por OMNeT++, en el fichero. Se puede ver un ejemplo en la Ilustración 46.

A continuación, como se puede observar en la Ilustración 48, se exponen los Rank de todos los racks que forman parte del escenario. Para este caso se trata de dos racks con las siguientes características:

- 1 rack con 2 Boards y 2 nodos por Board.
- 1 rack con 4 Boards y 4 nodos por Board.

El rango total son 20, 4 corresponden al primer rack y 16 al segundo.

En la última parte del fichero se asignarán las direcciones MAC a los elementos que forman parte del escenario. Para cada rack se asignará una dirección MAC por cada nodo contenido en el *Board*, de la misma manera que en el ejemplo anterior con el rango. Además se le asignará una dirección a cada *Board*, de cada rack.

Finalmente se asignará una dirección MAC a todos los nodos que forman el escenario, diferenciando si se tratan de nodos sencillos o de almacenamiento.

Para los switches se asignará una dirección por cada conexión establecida. Por cada nodo se añade una dirección, sin embargo en el caso de los racks se añade una dirección MAC por cada *Board* que tenga el rack. En el ejemplo de la Ilustración 49 para el switch[0] se han establecido tres conexiones, una por cada uno de los tres nodos conectados. En el caso correspondiente al switch[1] se han conectado 2 racks. Uno con 2 *Boards* y el otro con 4. En total se han asignado 6 direcciones MAC para el switch[1].

5.8.2 Scenario.ned

El otro fichero generado es el scenario.ned. En este fichero se escribirán todas las conexiones establecidas en el escenario y recogerá la información referente a la topología de red.

En primer lugar se definirán los nombres de los nodos de la plataforma SIMCAN a importar por parte del simulador.

A continuación se especificarán los canales de comunicación. Todas las conexiones que el usuario defina en el escenario, se enumerarán como canales. Además cada canal contendrá los parámetros de conexión previamente definidos en las conexiones:

- Delay
- Datarate

Posteriormente se escribirá en el fichero el nombre del escenario creado por el usuario. Irá seguido de los submódulos que lo integran como el de la configuración de la red. Se definirá por el tipo de simulación (secuencial o paralelo) y se escribirán los parámetros de red:

- Numboards (sólo se escribirá en la simulación en paralelo)
- Module Types (Tipos de módulos SIMCAN)
- Non IP Module Types (Tipos de módulos SIMCAN sin IP)
- Network address (Dirección por defecto de red)
- Net Mask (Máscara de red por defecto)

El resto de submódulos son los nodos simples, los racks y los switches definidos en el escenario.

En la definición de los nodos simples se escribirá el nombre de cada nodo acompañado del tipo de módulo que variará en función si está conectado o no a un swtich.

En el apartado de los racks se detallará el nombre y los parámetros que lo definen, incluyendo las conexiones de entrada y de salida (gatesizes in y gatesizes out).

Los switches se definirán por el nombre establecido por el usuario, el módulo y las conexiones de entrada y de salida (gatesizes in y gatesizes out).

Después de la definición de los submódulos se detallarán todas las conexiones establecidas en el escenario. Se escribirá el elemento inicial (nodo, rack o switch

que ya se ha descrito en la parte de submódulos) el canal definido previamente por el que se establecerá la conexión, y el elemento final.

Se seguirá la notación del simulador OMNET para definir la conexión, incluyendo la dirección y se especificará por donde se realiza las conexiones de entrada y de salida en cada elemento.

Finalmente se escribirá la instancia de la red., que contendrá el nombre de la red y el del escenario.

A continuación se detallará por apartados la estructura del fichero.

Al comienzo del fichero se especificarán los módulos a importar. A continuación se definirán las conexiones que se han establecido en el escenario, designándoles un nombre unívoco. A su vez se escribirán los parámetros de la conexión, como son el tiempo de retardo (*delay*) y la tasa de transferencia (*datarate*). Se puede ver un ejemplo referente a la cabecera de un fichero scenario.ned en la Ilustración 50.

En el siguiente paso, que se puede contemplar en la Ilustración 51, se define la configuración de red. Se escriben los parámetros generales y si se tratará de una simulación en secuencial o en paralelo. A continuación a cada nodo se le asignará un módulo de red en función si está conectado a o no a un switch.

En el caso de los racks se detallarán los parámetros que lo definen y los puertos de entrada y salida de los que dispone (*gatesize*). Serán tantos puertos de entrada y salida como *boards* disponga.

Para los switches sólo se definirán los puertos de entrada y salida, que serán tantos como conexiones establecidas tengan. Hay que tener en cuenta que el número de conexiones con un rack estará determinado con el número de *boards* de los que disponga.

Finalmente se escribirá el detalle de las conexiones establecidas con los switches. Las conexiones de los nodos se representarán con sus correspondientes direcciones, una de entrada y otra de salida. Las conexiones entre switches se realizarán de forma análoga.

En el caso de los racks habrá se escribirá una de entrada y otra de salida por cada *Board* que tenga el rack. Se recogen este detalle en el ejemplo expuesto en la Ilustración 52.

6 CONCLUSIONES Y TRABAJO FUTURO

Una vez alcanzado este punto, el desarrollo del proyecto ya culminado, se pueden obtener conclusiones de los resultados obtenidos y del periodo de realización del presente trabajo.

Como todo comienzo hubo que realizar un esfuerzo inicial para asumir globalmente las necesidades y los requisitos necesarios para abordar el proyecto. Un punto clave fue la asimilación de la estructura de la plataforma de simulación de SIMCAN. Había que comprender su cometido, las posibilidades que ofrecía y obtener de una manera eficiente la información contenida para integrarla en el presente proyecto.

El desarrollo del analizador sintáctico consumió más tiempo del esperado. Por una parte había que obtener toda la información necesaria de todos los ficheros y por otra había que almacenarla de tal manera que resultara viable su utilización.

Asimismo el diseño de la interfaz de usuario requirió un esfuerzo extra de tiempo. El hecho de realizar la configuración de los elementos dinámicamente, en función de los datos obtenidos de SIMCAN, afecta directamente a las pantallas que se muestran al usuario. Estas pantallas se generan de manera dinámica en vez de estática en un diseñador gráfico y todo el tratamiento de la información como la obtención de datos y validaciones necesariamente deben tratarse de manera dinámica. Esta funcionalidad condicionó la estructura lógica de las clases afectadas para acometer su implementación.

La información parcial que introduce el usuario cuando configura elementos, se ha de estructurar de tal forma que cuando complete la información del

elemento sea viable su posterior tratamiento. Por esta razón se eligió la utilización de ficheros xml para poder abordar de manera flexible y dinámica la información.

Por otra parte, el desarrollo del resto de la aplicación, como la gestión de escenarios y la generación de los ficheros de salida, fue más ágil y requirió menos tiempo que la parte anterior.

Finalmente, al realizar un repaso a todos los puntos necesarios para desarrollar el presente trabajo y al estudiar los resultados obtenidos en las pruebas realizadas, se llega a comprender el servicio que aporta la presente aplicación.

Pasar de una generación manual de ficheros a una forma automática permite la creación de entornos de simulación distribuidos a gran escala de una manera cómoda y eficiente. No existe la necesidad de tener que repasar la sintaxis de los ficheros, ni corregir pequeños errores que arruinan simulaciones que se ejecutan durante varias horas con el coste de recursos que eso supone.

A partir del trabajo actual podrían realizarse una serie mejorar orientadas a las siguientes facetas:

- Dado que la aplicación ofrece la opción de ejecutar un entorno de simulación, se podría añadir un planificador que ejecutara distintos entornos en las horas añadidas por el usuario. Hay simulaciones que tardan muchas horas en ejecutarse, se podrían ejecutar varias seguidas sin tener que estar presencialmente, sin tener que esperar a que finalice la anterior.

- Se pueden introducir mejoras en la interfaz gráfica de la aplicación que ofrezcan al usuario más opciones de personalización y de configuración. Tanto de los elementos del escenario como de los repositorios de ficheros.

ANEXO I. PRESUPUESTO

En esta sección se va a proceder a detallar el presupuesto total necesario para la consecución del proyecto. En este presupuesto se incluyen todos los gastos necesarios para su total elaboración y puesta en marcha.

TAREA	HORAS DEDICADAS
Análisis	80 h
Diseño	120 h
Implementación	340 h
Pruebas	60 h
Documentación	200 h
Total	800 h

Tabla 39 Desglose por actividades del proyecto

La tabla anterior recoge las distintas actividades que se llevaron a cabo para la realización del presente proyecto. Cada actividad lleva asociadas el número de horas que fueron necesarias para su elaboración.

Puede observarse que la actividad que más tiempo precisó fue la implementación del código del sistema, que supuso casi el 45% del tiempo total invertido.

Para la realización de la documentación también se requirió un tiempo sensiblemente superior al resto de las actividades. Junto a la implementación suponen el 70% del tiempo total.

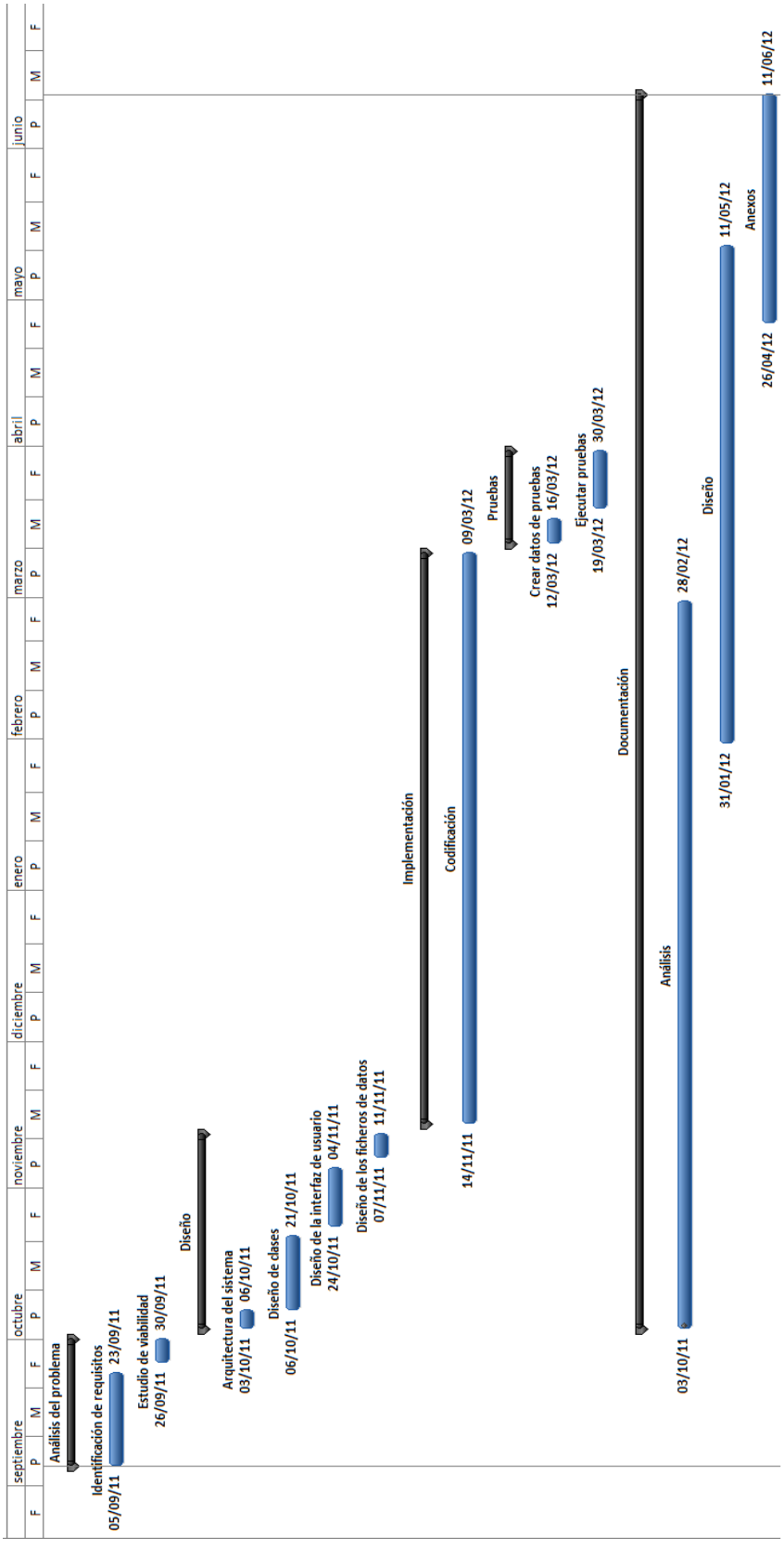

Ilustración 38 Diagrama de Gantt

En el diagrama de Gantt de tareas expuesto en la Ilustración 38 se puede apreciar extensión de las actividades y su duración. La documentación se llevó a cabo en paralelo al resto de actividades, siendo ésta secundaria., a excepción de la fase final que tuvo dedicación exclusiva.

Si se toma como referencia el tiempo total invertido en la elaboración del presente proyecto, 800 horas, y teniendo en cuenta que:

- La jornada laboral ha sido de 4 horas diarias
- Cada mes consta de 20 días laborables

Se llega a la conclusión que el periodo de tiempo necesario en la elaboración del proyecto ha sido aproximadamente de 10 meses.

Para la realización del proyecto se requiere personal informático cualificado, capaz de llevar a cabo las tareas reflejadas en la tabla anterior. Por esta razón será necesario que el personal adopte su correspondiente rol, los cuales se recogen en la tabla que aparece a continuación.

CARGO	SUELDO NETO	SUELDO BRUTO	COSTE/HORA
Analista	1.532 €/mes	28.000 €/año	25€
Diseñador	1.532 €/mes	28.000 €/año	25€
Programador	1.225,6 €/mes	22.400 €/año	20€
Responsable de pruebas	1.225,6 €/mes	22.400 €/año	20€
Responsable de documentación	1.225,6 €/mes	22.400 €/año	20€

Tabla 40 Salarios por categoría

En la Tabla anterior se recogen los salarios neto y bruto correspondientes a los distintos roles que el personal del proyecto deberá adoptar para la realización del mismo. En dicha tabla la información se divide en:

- Coste / Hora: indica el sueldo bruto en una hora de trabajo.
- Sueldo Bruto: indica el sueldo bruto anual, con 14 pagas mensuales.
- Sueldo Neto: indica el sueldo neto mensual. Se descuenta el I.R.P.F (17%) y Seguridad Social (6.4%)

Los gastos de personal imputables al proyecto se basan en el personal encargado del desarrollo del proyecto. En este caso compuesto por un informático, ha necesitado adoptar todos y cada uno de los distintos roles especificados en la Tabla 40 para completar las actividades anunciadas anteriormente (Tabla 39).

La siguiente tabla recoge el coste total de personal asociado al proyecto.

CARGO	PERSONAL	HORAS	COSTE/H	TOTAL
Analista	1	80 h	25€	2000€
Diseñador	1	120 h	25€	3000€
Programador	1	340 h	20€	6800€
Responsable de pruebas	1	60 h	20€	1200€
Responsable de documentación	1	200 h	20€	4000€
TOTAL		800 h		17000€

Tabla 41 Coste total personal

Según la información contenida en la Tabla 41, los gastos referentes al personal encargado del desarrollo del proyecto ascienden a **17.000 €**

Una vez expuestos los gastos personales, a continuación se detallarán los gastos referentes a los recursos materiales necesarios para poder llevar a cabo la elaboración del presente proyecto.

RECURSO	CANTIDAD	COSTE TOTAL
Ordenador Personal	1	600€
Router	1	60€
Cable RJ45	1	25€
Microsoft Windows 7	1	319€
Microsoft Office 2007	1	200€
Ubuntu 10.04	1	0€
NeatBeans 6.9	1	0€
Total		1204€

Tabla 42 Recursos materiales empleados

Para calcular el presupuesto total del proyecto hay que añadir a los gastos del personal y a los recursos materiales los gastos indirectos derivados, el margen de imprevistos y el beneficio a obtener. El desglose por detalle se muestra en la siguiente tabla:

DESCRIPCIÓN	COSTE
Personal con cargo al proyecto	17.000,00 €
Recursos materiales empleados	1.204,00 €
Gastos indirectos	1.700,00 €
Total gastos	19.904,00 €
Margen de imprevistos (10%)	1.990,40 €
Margen de beneficio (15%)	2.985,60 €
Subtotal	24.880,00 €
I.V.A. (18%)	4.478,40 €

TOTAL	**29.358,40 €**

Tabla 43 Resumen del presupuesto

Los gastos indirectos se obtienen a partir de los gastos derivados del personal al cargo de desarrollar el proyecto, se estiman en un 10% de estos gastos. También se ha contemplado un margen de imprevistos que suponen un 10% de las suma de todos los costes: personal, recursos materiales y gastos indirectos.

Finalmente se aplica sobres los gastos totales un 15% de margen de beneficio para la empresa y se imputa al total el I.V.A en vigor del 18%, como se detalla en la Tabla 43.

El presupuesto total del proyecto asciende a **29.358,40 €** I.V.A. incluido.

ANEXO II. CASO PRÁCTICO

En este apartado se va mostrar un ejemplo práctico de generación de un escenario. Si bien el sistema es multiplataforma, ya que se ha realizado mediante el lenguaje de programación Java, se ha optado por generación del escenario mediante el sistema operativo Windows por simplicidad a la hora de integrarlo en la documentación.

La primera acción a llevar a cabo es generar el fichero del sistema que recopila la información relevante de la plataforma de simulación SIMCAN. Sólo es necesario seleccionar la opción "*Scan Directory*" en el menú "*Tools*" y escribir la ruta donde se encuentra el repositorio SIMCAN, como puede verse en la Ilustración 39.

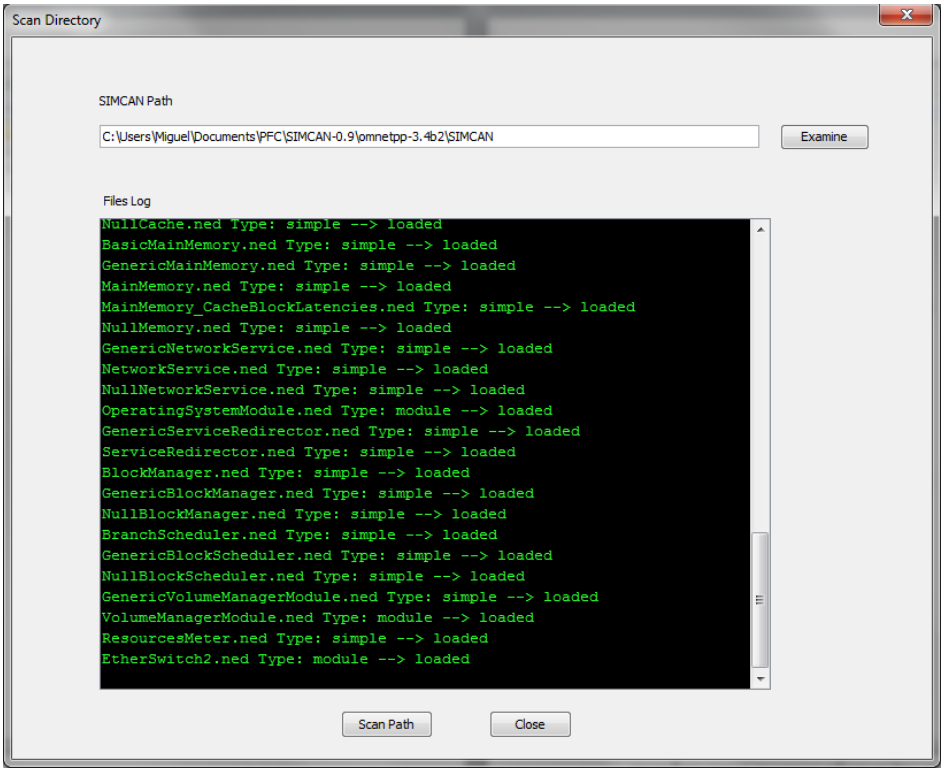

Ilustración 39 Pantalla de generación del fichero del sistema

Al finalizar aparecerá un log de los módulos que se han añadido al fichero del sistema. En la Ilustración 42 se muestra un ejemplo de un módulo complejo y debajo otro simple con la información obtenida de la plataforma SIMCAN.

El siguiente paso es crear un nuevo escenario. En el menú "*Scenarios*" se seleccionará la opción "*New Scenario*". Y en la ventana emergente se escribirá el nombre del nuevo escenario.

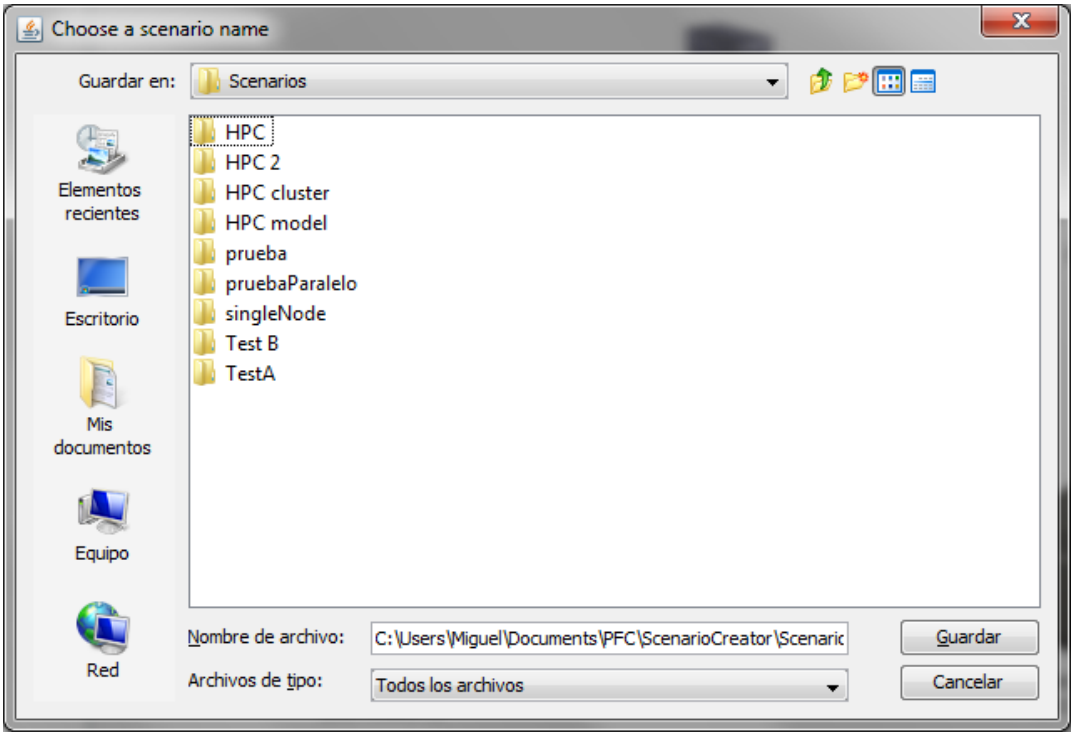

Ilustración 40 Pantalla crear nuevo escenario

Cuando se crea un escenario se tiene la opción añadir los elementos ya creados en el árbol de elementos según su tipo, bien sean: nodos, switches o racks. También se pueden crear nuevos elementos para poder añadirlos posteriormente.

Para este ejemplo se cargarán 6 nodos. Cuatro nodos (Dual core node) serán del mismo tipo, y los dos restantes serán de tipos distintos entre sí. Para añadir varios nodos del mismo tipo se ha de seleccionar la opción "*Add n nodes to scenario*" en vez de "*Add node to scenario*" al seleccionar el nodo en el árbol de

elementos. Los nodos almacenados se cargarán de su correspondiente fichero XML, que contiene toda la información. Se puede ver cómo se almacena el sistema operativo de un nodo en la Ilustración 43.

Además se añadirán 2 switches de distintas características, y 2 racks. Con las siguientes características generales:

- 1 rack con 2 Boards y 2 nodos por Board.
- 1 rack con 4 Boards y 4 nodos por Board.

Se conectarán los nodos con un swtich, los racks con el otro switch y la última conexión existente será entre los switches, como puede apreciarse en la Ilustración 41.

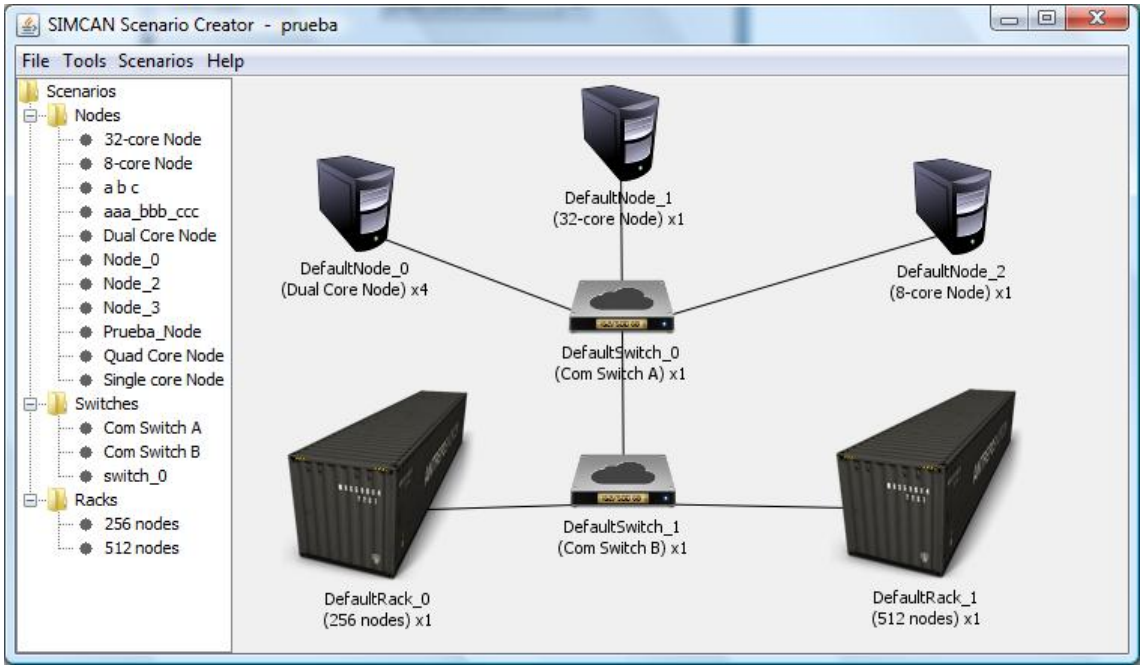

Ilustración 41 Pantalla de escenario caso práctico

Una vez que ya tengamos el escenario creado se podrán generar los ficheros de salida "*omnet.ini*" y "*scenario.ned*" mediante el menú "*Scenarios*" y la opción "*Generate config files*".

Ficheros generados

```
<Module Dir_Type="OperatingSystems">
  <Name>OperatingSystemModule</Name>
  <Type>module</Type>
  <Parameter Comment=" Node's IP" Name="localIP" Type="string"/>
  <Parameter Comment=" Service Redirector
   type" Name="serviceRedirectorType" Type="string"/>
  <Parameter Comment=" CPU Service
   type" Name="cpuSchedulerType" Type="string"/>
  <Parameter Comment=" Cache type" Name="memoryType" Type="string"/>
  <Parameter Comment=" File System module
   type" Name="fsModuleType" Type="string"/>
  <Parameter Comment=" Volume Manager module
   type" Name="vmModuleType" Type="string"/>
  <Parameter Comment=" Number of applications"
   Name="numApps" Type="numeric const"/>
  <Parameter Comment=" Number of CPUs" Name="numCPUs" Type="numeric
   const"/>
  <Parameter Comment=" Number of File Systems" Name="
   numFS" Type="numeric const"/>
  <Parameter Comment=" Number of Servers"
   Block Name="numBlockServers" Type="numeric const"/>
  <Gate Name="fromApps[]" Type="in"/>
  <Gate Name="fromCPU[]" Type="in"/>
  <Gate Name="toApps[]" Type="out"/>
  <Gate Name="toCPU[]" Type="out"/>
  <Submodule>
   <Subname>serviceRedirector</Subname>
   <Subtype>serviceRedirectorType like GenericServiceRedirector
   </Subtype>
   <Subparameter Name="numApps" Value="numApps"/>
   <Gatesize>fromApps[numApps]</Gatesize>
   <Gatesize>toApps[numApps]</Gatesize>
  </Submodule>
 </Module>
  <Module Dir_Type="ServiceRedirectors">
  <Name>ServiceRedirector</Name>
  <Type>simple</Type>
  <Parameter Comment=" Number of
   applications" Name="numApps" Type="numeric const"/>
  <Gate Name="fromApps[]" Type="in"/>
  <Gate Name="fromMemory" Type="in"/>
  <Gate Name="fromCPU" Type="in"/>
  <Gate Name="toApps[]" Type="out"/>
  <Gate Name="toMemory" Type="out"/>
  <Gate Name="toCPU" Type="out"/>
 </Module>
```

Ilustración 42 Fichero del sistema

```
<OS>
 <serviceRedirectorType Type="ServiceRedirector">
  <Parameter Name="numApps" Type="numeric const" Value="2"/>
 </serviceRedirectorType>
 <networkServiceType Type="NullNetworkService">
  <Parameter Name="localIP" Type="string" Value="192.2"/>
 </networkServiceType>
 <cpuSchedulerType Type="CPU_Scheduler_RR">
  <Parameter Name="numCPUs" Type="numeric const" Value="3"/>
  <Parameter Name="quantum" Type="numeric const" Value="2"/>
 </cpuSchedulerType>
 <memoryType Type="MainMemory">
  <Parameter Name="readLatencyTime" Type="numeric const" Value="2"/>
  <Parameter Name="writeLatencyTime" Type="numeric
   const" Value="2"/>
  <Parameter Name="flushTime_s" Type="numeric const" Value="2"/>
  <Parameter Name="size_KB" Type="numeric const" Value="2"/>
  <Parameter Name="sizeCache_KB" Type="numeric const" Value="2"/>
  <Parameter Name="blockSize_KB" Type="numeric const" Value="2"/>
  <Parameter Name="readAheadBlocks" Type="numeric const" Value="2"/>
  <Parameter Name="numInputs" Type="numeric const" Value="1"/>
 </memoryType>
 <vmModuleType Type="VMModule">
  <blockManagerType Type="NullBlockManager">
   <Subparameter Name="numBlockServers" Type="numeric const"
    Value="2"/>
  </blockManagerType>
  <schedulerType Type="BranchScheduler"/>
  <cacheType Type="NullCache">
   <Subparameter Name="numInputs" Type="numeric const" Value="2"/>
  </cacheType>
  <Parameter Name="numFS" Type="numeric const" Value="2"/>
  <Parameter Name="numBlockServers" Type="numeric const" Value="2"/>
 </vmModuleType>
 <fsModuleType Type="FSModule">
  <FSROW BlockSize="0" FSType="ext2" FillRatio="0.0" Index="0"
   NumBlocks="0" PreloadFile="" VFSPath="/cana/remote"/>
  <FSROW BlockSize="0" FSType="ext2" FillRatio="0.0" Index="1"
   NumBlocks="0" PreloadFile="" VFSPath="/lukas"/>
 </fsModuleType>
 <OSParameter Name="localIP" Type="string" Value="192.2"/>
 <OSParameter Name="numApps" Type="numeric const" Value="2"/>
 <OSParameter Name="numCPUs" Type="numeric const" Value="3"/>
 <OSParameter Name="numFS" Type="numeric const" Value="2"/>
 <OSParameter Name="numBlockServers" Type="numeric const"
  Value="2"/>
</OS>
```

Ilustración 43 Detalle del SO de un nodo

En fichero *omnet.ini* se recogerá la información de todos los elementos con el formato apropiado para poder ser procesado por el simulador OMNeT++, como se expuso en el apartado 5.8.1 Omnet.ini.

```
[General]
preload-ned-files = *.ned @/home/user/applics/omnetpp-3.4b2_old/SIMCAN/nedfiles.lst
@/home/user/applics/omnetpp-3.4b2_old/SIMCAN/../INET-20061020/nedfiles.lst
parallel-simulation=false
## Enable_parallel_simulation
network=prueba
## Network's name
total-stack-kb=7535
##############################################################################
#
[OutVectors]
**.enabled = no
## Disable output vectors
##############################################################################
#
[Cmdenv]
express-mode = yes
## Express mode (no messages)
##############################################################################
#
[Tkenv]
plugin-path=../../../Etc/plugins
default-run=1
##############################################################################
```

Ilustración 44 Cabecera del fichero de salida omnet.ini

Una vez escrita la cabecera se escribe la configuración de los elementos. Para el caso de los nodos se detalla como se recoge en la Ilustración 45.

```
[Parameters]
##################################################################
###     Definition of node DefaultNode_1
##################################################################
prueba.DefaultNode_1.hostName = "DefaultNode_1";
prueba.DefaultNode_1.numApps = 2;
prueba.DefaultNode_1.numCPUs = 3;
prueba.DefaultNode_1.numFS = 2;
prueba.DefaultNode_1.numBlockServers = 2;
prueba.DefaultNode_1.appModuleType = "AppModule";
prueba.DefaultNode_1.cpuModuleType = "CPU_Module";
prueba.DefaultNode_1.osModuleType = "OperatingSystemModule";
prueba.DefaultNode_1.bsModuleType = "BlockServerModule";

 ### Application_0 -> TracePlayer_Sequential
 prueba.DefaultNode_1.appModule[0].appType = "TracePlayer_Sequential";
 prueba.DefaultNode_1.appModule[0].app.application_netType = "1";
 prueba.DefaultNode_1.appModule[0].app.startDelay = 2;
 prueba.DefaultNode_1.appModule[0].app.traceFile = "3";
 prueba.DefaultNode_1.appModule[0].app.timesFile = "1";

 ### CPU
 prueba.DefaultNode_1.cpuModule.cpuCoreType = "CPUcore";
 prueba.DefaultNode_1.cpuModule.numCores = 1;
 prueba.DefaultNode_1.cpuModule.speed = 2;

 ### Block servers
 prueba.DefaultNode_1.bsModule[*].diskType = "Disk_400GB_LI";
 prueba.DefaultNode_1.bsModule[*].cacheType = "NullCache";

 ### Operating System
 prueba.DefaultNode_1.osModule.serviceRedirectorType = "ServiceRedirector";
 prueba.DefaultNode_1.osModule.serviceRedirector.numApps = 2;
 prueba.DefaultNode_1.osModule.networkServiceType = "NullNetworkService";
 prueba.DefaultNode_1.osModule.cpuSchedulerType = "CPU_Scheduler_RR";
```

```
prueba.DefaultNode_1.osModule.cpuScheduler.numCPUs = 3;
 prueba.DefaultNode_1.osModule.cpuScheduler.quantum = 2;
 prueba.DefaultNode_1.osModule.memoryType = "MainMemory";
 prueba.DefaultNode_1.osModule.ioRedirectorType = "IORedirector";
 prueba.DefaultNode_1.osModule.vmModuleType = "VMModule";
 prueba.DefaultNode_1.osModule.fsModuleType = "FSModule";
 prueba.DefaultNode_1.osModule.numApps = 2;
 prueba.DefaultNode_1.osModule.numCPUs = 3;
 prueba.DefaultNode_1.osModule.numFS = 2;
 prueba.DefaultNode_1.osModule.numBlockServers = 2;

  ### Memory
  prueba.DefaultNode_1.osModule.memory.readLatencyTime_s = 2;
  prueba.DefaultNode_1.osModule.memory.writeLatencyTime_s = 2;
  prueba.DefaultNode_1.osModule.memory.flushTime_s = 2;
  prueba.DefaultNode_1.osModule.memory.size_KB = 2;
  prueba.DefaultNode_1.osModule.memory.sizeCache_KB = 2;
  prueba.DefaultNode_1.osModule.memory.blockSize_KB = 2;
  prueba.DefaultNode_1.osModule.memory.readAheadBlocks = 2;
  prueba.DefaultNode_1.osModule.memory.numInputs = 1;

  ### I/O Redirector
  prueba.DefaultNode_1.osModule.ioRedirector.nfsClientIndex = 0;

  ### Volume Manager
  prueba.DefaultNode_1.osModule.vmModule.blockManagerType = "NullBlockManager";
  prueba.DefaultNode_1.osModule.vmModule.blockManager.numBlockServers = 2;
  prueba.DefaultNode_1.osModule.vmModule.schedulerType = "BranchScheduler";
  prueba.DefaultNode_1.osModule.vmModule.cacheType = "NullCache";
  prueba.DefaultNode_1.osModule.vmModule.cache.numInputs = 2;
```

Ilustración 45 Definición de un elemento nodo en el fichero omnet.ini

A continuación el detalle del rack con con 2 Boards y 2 nodos por Board, como se puede observar en la Ilustración 46.

```
##################################################################
###    Definition of rack DefaultRack_0
##################################################################
prueba.DefaultRack_0.numBoards = 2;
prueba.DefaultRack_0.nodesPerBoard = 2;
prueba.DefaultRack_0.NodeType = "Quad Core Node";
##################################################################
###    Definition of node Quad Core Node
##################################################################
prueba.DefaultRack_0.Quad Core Node[*].hostName = "Quad Core Node";
prueba.DefaultRack_0.Quad Core Node[*].numApps = 2;
prueba.DefaultRack_0.Quad Core Node[*].numCPUs = 3;
prueba.DefaultRack_0.Quad Core Node[*].numFS = 2;
prueba.DefaultRack_0.Quad Core Node[*].numBlockServers = 2;
prueba.DefaultRack_0.Quad Core Node[*].appModuleType = "AppModule";
prueba.DefaultRack_0.Quad Core Node[*].cpuModuleType = "CPU_Module";
prueba.DefaultRack_0.Quad Core Node[*].osModuleType = "OperatingSystemModule";
prueba.DefaultRack_0.Quad Core Node[*].bsModuleType = "BlockServerModule";

 ### Application_0 -> TracePlayer_Sequential
 prueba.DefaultRack_0.Quad Core Node[*].appModule[0].appType = "TracePlayer_Sequential";
 prueba.DefaultRack_0.Quad Core Node[*].appModule[0].app.application_netType = "1";
 prueba.DefaultRack_0.Quad Core Node[*].appModule[0].app.startDelay = 2;
 prueba.DefaultRack_0.Quad Core Node[*].appModule[0].app.traceFile = "3";
 prueba.DefaultRack_0.Quad Core Node[*].appModule[0].app.timesFile = "1";
```

Ilustración 46 Definición de un elemento rack en el fichero omnet.ini

La parte que contiene los elementos finaliza con los switches. En estos casos se ha mostrado un solo elemento por tipo, en vez del detalle de todos los elementos para facilitar el seguimiento. En la Ilustración 47 se puede ver el detalle de un switch.

```
##################################################################
###    Definition of switch DefaultSwitch_0
##################################################################

prueba.DefaultSwitch_0.relayUnit.relayUnitType = "MACRelayUnitPP";
   prueba.DefaultSwitch_0.relayUnit.agingTime = "54";
   prueba.DefaultSwitch_0.relayUnit.processingTime = "664";
   prueba.DefaultSwitch_0.relayUnit.bufferSizeb = "343";
   prueba.DefaultSwitch_0.relayUnit.highWatermark = "434";
```

Ilustración 47 Definición de un elemento switch en el fichero omnet.ini

En la siguiente sección del fichero omnet.ini se asigna un Rank a cada nodo de los racks. El primer rack (rack [0]) le corresponden 4 asignaciones al contener 2 boards y 2 nodos por board. Al segundo rack (rack [1]) le corresponden 16 al tratarse de un rack con 4 boards y 4 nodos por board. El detalle se puede observar en la Ilustración 48.

Finalmente el fichero *omnet.ini* contendrá las direcciones MAC asignadas a los elementos que forman parte del escenario, de la siguiente manera:

- 20 direcciones a los nodos contenidos en los racks (4 +16)
- 6 direcciones, una correspondiente a cada uno de los 6 nodos.
- 26 direcciones a los board de los dos racks (6 + 20). Uno por el board y otro para cada nodo.
- 14 direcciones a los switches (7 + 7)
    - El primer switch: 6 nodos y la conexión al switch
    - El segundo switch: 2 boards + 4 boards + conexión al otro switch

```
### Rank for each application
prueba.rack[0].nodeBoard[0].node[0].appModule[1].app.myRank = 0;
prueba.rack[0].nodeBoard[0].node[1].appModule[1].app.myRank = 1;
prueba.rack[0].nodeBoard[1].node[0].appModule[1].app.myRank = 2;
prueba.rack[0].nodeBoard[1].node[1].appModule[1].app.myRank = 3;
prueba.rack[1].nodeBoard[0].node[0].appModule[1].app.myRank = 4;
prueba.rack[1].nodeBoard[0].node[1].appModule[1].app.myRank = 5;
prueba.rack[1].nodeBoard[0].node[2].appModule[1].app.myRank = 6;
prueba.rack[1].nodeBoard[0].node[3].appModule[1].app.myRank = 7;
prueba.rack[1].nodeBoard[1].node[0].appModule[1].app.myRank = 8;
prueba.rack[1].nodeBoard[1].node[1].appModule[1].app.myRank = 9;
prueba.rack[1].nodeBoard[1].node[2].appModule[1].app.myRank = 10;
prueba.rack[1].nodeBoard[1].node[3].appModule[1].app.myRank = 11;
prueba.rack[1].nodeBoard[2].node[0].appModule[1].app.myRank = 12;
prueba.rack[1].nodeBoard[2].node[1].appModule[1].app.myRank = 13;
prueba.rack[1].nodeBoard[2].node[2].appModule[1].app.myRank = 14;
prueba.rack[1].nodeBoard[2].node[3].appModule[1].app.myRank = 15;
prueba.rack[1].nodeBoard[3].node[0].appModule[1].app.myRank = 16;
prueba.rack[1].nodeBoard[3].node[1].appModule[1].app.myRank = 17;
prueba.rack[1].nodeBoard[3].node[2].appModule[1].app.myRank = 18;
prueba.rack[1].nodeBoard[3].node[3].appModule[1].app.myRank = 19;
```

Ilustración 48 Ranks de los racks en el fichero omnet.ini

El detalle de las direcciones MAC asignadas puede verse en la Ilustración 49 que se divide en tres páginas.

```
### MAC addresses for nodes in rack[0]
prueba.rack[0].nodeBoard[0].node[0].eth[*].mac.address = "0000AA000108";
prueba.rack[0].nodeBoard[0].node[1].eth[*].mac.address = "0000AA000109";
prueba.rack[0].nodeBoard[1].node[0].eth[*].mac.address = "0000AA00010A";
prueba.rack[0].nodeBoard[1].node[1].eth[*].mac.address = "0000AA00010B";
### MAC addresses for nodes in rack[1]
prueba.rack[1].nodeBoard[0].node[0].eth[*].mac.address = "0000AA00010C";
prueba.rack[1].nodeBoard[0].node[1].eth[*].mac.address = "0000AA00010D";
prueba.rack[1].nodeBoard[0].node[2].eth[*].mac.address = "0000AA00010E";
prueba.rack[1].nodeBoard[0].node[3].eth[*].mac.address = "0000AA00010F";
prueba.rack[1].nodeBoard[1].node[0].eth[*].mac.address = "0000AA000110";
prueba.rack[1].nodeBoard[1].node[1].eth[*].mac.address = "0000AA000111";
prueba.rack[1].nodeBoard[1].node[2].eth[*].mac.address = "0000AA000112";
prueba.rack[1].nodeBoard[1].node[3].eth[*].mac.address = "0000AA000113";
prueba.rack[1].nodeBoard[2].node[0].eth[*].mac.address = "0000AA000114";
prueba.rack[1].nodeBoard[2].node[1].eth[*].mac.address = "0000AA000115";
prueba.rack[1].nodeBoard[2].node[2].eth[*].mac.address = "0000AA000116";
prueba.rack[1].nodeBoard[2].node[3].eth[*].mac.address = "0000AA000117";
prueba.rack[1].nodeBoard[3].node[0].eth[*].mac.address = "0000AA000118";
prueba.rack[1].nodeBoard[3].node[1].eth[*].mac.address = "0000AA000119";
prueba.rack[1].nodeBoard[3].node[2].eth[*].mac.address = "0000AA00011A";
prueba.rack[1].nodeBoard[3].node[3].eth[*].mac.address = "0000AA00011B";

prueba.storage[0].eth[*].mac.address = "0000AA00011C";
prueba.singleNode[0].eth[*].mac.address = "0000AA00011D";
prueba.singleNode[1].eth[*].mac.address = "0000AA00011E";
prueba.singleNode[2].eth[*].mac.address = "0000AA00011F";
prueba.singleNode[3].eth[*].mac.address = "0000AA000120";
prueba.singleNode[4].eth[*].mac.address = "0000AA000121";
### MAC addresses for rack[0].board[0]
prueba.rack[0].].nodeBoard[0].switch.mac[0].address = "0000AA000122";
prueba.rack[0].].nodeBoard[0].switch.mac[1].address = "0000AA000123";
prueba.rack[0].].nodeBoard[0].switch.mac[2].address = "0000AA000124";
```

```
### MAC addresses for rack[0].board[1]
prueba.rack[0].].nodeBoard[1].switch.mac[0].address = "0000AA000125";
prueba.rack[0].].nodeBoard[1].switch.mac[1].address = "0000AA000126";
prueba.rack[0].].nodeBoard[1].switch.mac[2].address = "0000AA000127";
### MAC addresses for rack[1].board[0]

prueba.rack[1].].nodeBoard[0].switch.mac[0].address = "0000AA000128";
prueba.rack[1].].nodeBoard[0].switch.mac[1].address = "0000AA000129";
prueba.rack[1].].nodeBoard[0].switch.mac[2].address = "0000AA00012A";
prueba.rack[1].].nodeBoard[0].switch.mac[3].address = "0000AA00012B";
prueba.rack[1].].nodeBoard[0].switch.mac[4].address = "0000AA00012C";

### MAC addresses for rack[1].board[1]
prueba.rack[1].].nodeBoard[1].switch.mac[0].address = "0000AA00012D";
prueba.rack[1].].nodeBoard[1].switch.mac[1].address = "0000AA00012E";
prueba.rack[1].].nodeBoard[1].switch.mac[2].address = "0000AA00012F";
prueba.rack[1].].nodeBoard[1].switch.mac[3].address = "0000AA000130";
prueba.rack[1].].nodeBoard[1].switch.mac[4].address = "0000AA000131";

### MAC addresses for rack[1].board[2]
prueba.rack[1].].nodeBoard[2].switch.mac[0].address = "0000AA000132";
prueba.rack[1].].nodeBoard[2].switch.mac[1].address = "0000AA000133";
prueba.rack[1].].nodeBoard[2].switch.mac[2].address = "0000AA000134";
prueba.rack[1].].nodeBoard[2].switch.mac[3].address = "0000AA000135";
prueba.rack[1].].nodeBoard[2].switch.mac[4].address = "0000AA000136";

### MAC addresses for rack[1].board[3]
prueba.rack[1].].nodeBoard[3].switch.mac[0].address = "0000AA000137";
prueba.rack[1].].nodeBoard[3].switch.mac[1].address = "0000AA000138";
prueba.rack[1].].nodeBoard[3].switch.mac[2].address = "0000AA000139";
prueba.rack[1].].nodeBoard[3].switch.mac[3].address = "0000AA00013A";
prueba.rack[1].].nodeBoard[3].switch.mac[4].address = "0000AA00013B";
```

```
### MAC addresses for switchRack[0]
**.Com Switch A[0].mac[0].address = "0000AA00013C";
**.Com Switch A[0].mac[1].address = "0000AA00013D";
**.Com Switch A[0].mac[2].address = "0000AA00013E";
**.Com Switch A[0].mac[3].address = "0000AA00013F";
**.Com Switch A[0].mac[4].address = "0000AA000140";
**.Com Switch A[0].mac[5].address = "0000AA000141";
**.Com Switch A[0].mac[6].address = "0000AA000142";



### MAC addresses for switchRack[1]
**.Com Switch B[1].mac[0].address = "0000AA000143";
**.Com Switch B[1].mac[1].address = "0000AA000144";
**.Com Switch B[1].mac[2].address = "0000AA000145";
```

Ilustración 49 Asignación de direcciones MAC en el fichero omnet.ini

El fichero *scenario.ned* recoge todas las conexiones establecidas así como la topología de red como se ha explicado en el apartado 5.8.2 Scenario.ned. En las siguientes ilustraciones se expone la generación de este fichero para el escenario tratado en este caso práctico.

```
import
  "Node_without_Net",
  "Node_TCP",
// ------------------------------------------------------------
//     Definition of channels
// ------------------------------------------------------------
channelprueba_channel_0
  delay 0.0us;
  datarate 0*1000000;
endchannel

channelprueba_channel_1
  delay 0.0us;
  datarate 0*1000000;
endchannel

channelprueba_channel_2
  delay 0.0us;
  datarate 0*1000000;
endchannel

channelprueba_channel_3
  delay 0.0us;
  datarate 0*1000000;
endchannel

channelprueba_channel_4
  delay 0.0us;
  datarate 0*1000000;
endchannel

// ------------------------------------------------------------
//     Scenario name
// ------------------------------------------------------------
module Prueba

  // ------------------------------------------------------------
  //     Definition of main modules
  // ------------------------------------------------------------
  submodules:
```

Ilustración 50 Cabecera del fichero scenario.ned

```
// ------------------------------------------------------------------
//     Network configurator (sequential simulation)
// ------------------------------------------------------------------
netConfig: FlatNetworkConfigurator;
  parameters:
    moduleTypes = "Node_TCP Node_TCP_Config EtherSwitch2",
    nonIPModuleTypes = "EtherSwitch2",
    networkAddress = "192.168.0.0",
    netmask = "255.255.255.0";

// ------------------------------------------------------------------
//     Definition of single nodes
// ------------------------------------------------------------------
DefaultNode_0: Node_TCP_Config[4];
DefaultNode_1: Node_TCP_Config;
DefaultNode_2: Node_TCP_Config;

// ------------------------------------------------------------------
//     Definition of racks
// ------------------------------------------------------------------
DefaultRack_0: Rack;
  parameters:
    numBoards = 2,
    nodesPerBoard = 2,
    NodeType = Quad Core Node;
  gatesizes:
    in[2],
    out[2];
DefaultRack_1: Rack;
  parameters:
    numBoards = 4,
    nodesPerBoard = 4,
    NodeType = Quad Core Node;
  gatesizes:
    in[4],
    out[4];

// ------------------------------------------------------------------
//     Definition of switches
// ------------------------------------------------------------------
DefaultSwitch_0: EtherSwitch2;
  gatesizes:
    in[7],
    out[7];

DefaultSwitch_1: EtherSwitch2;
  gatesizes:
    in[7],
    out[7];
```

Ilustración 51 Configuración de red en el fichero scenario.ned

```
// -------------------------------------------------------------
  //    Connection between modules
  // -------------------------------------------------------------
  connections nocheck:

   // -------------------------------------------------------------
   //    Connection between DefaultNode_0 and DefaultSwitch_0
   // -------------------------------------------------------------
   DefaultNode_0[0].ethOut --> prueba_channel_0 --> DefaultSwitch_0.in[0];
   DefaultNode_0[0].ethIn <-- prueba_channel_0 <-- DefaultSwitch_0.out[0];
   DefaultNode_0[1].ethOut --> prueba_channel_0 --> DefaultSwitch_0.in[1];
   DefaultNode_0[1].ethIn <-- prueba_channel_0 <-- DefaultSwitch_0.out[1];
   DefaultNode_0[2].ethOut --> prueba_channel_0 --> DefaultSwitch_0.in[2];
   DefaultNode_0[2].ethIn <-- prueba_channel_0 <-- DefaultSwitch_0.out[2];
   DefaultNode_0[3].ethOut --> prueba_channel_0 --> DefaultSwitch_0.in[3];
   DefaultNode_0[3].ethIn <-- prueba_channel_0 <-- DefaultSwitch_0.out[3];

   // -------------------------------------------------------------
   //    Connection between DefaultNode_1 and DefaultSwitch_0
   // -------------------------------------------------------------
   DefaultNode_1.ethOut --> prueba_channel_1 --> DefaultSwitch_0.in[4];
   DefaultNode_1.ethIn <-- prueba_channel_1 <-- DefaultSwitch_0.out[4];

   // -------------------------------------------------------------
   //    Connection between DefaultNode_2 and DefaultSwitch_0
   // -------------------------------------------------------------
   DefaultNode_2.ethOut --> prueba_channel_2 --> DefaultSwitch_0.in[5];
   DefaultNode_2.ethIn <-- prueba_channel_2 <-- DefaultSwitch_0.out[5];

  // -------------------------------------------------------------
   //    Connection between DefaultRack_1 and DefaultSwitch_1
   // -------------------------------------------------------------
   DefaultRack_1.out[0] --> prueba_channel_3 --> DefaultSwitch_1.in[0];
   DefaultRack_1.in[0] <-- prueba_channel_3 <-- DefaultSwitch_1.out[0];
   DefaultRack_1.out[1] --> prueba_channel_3 --> DefaultSwitch_1.in[1];
   DefaultRack_1.in[1] <-- prueba_channel_3 <-- DefaultSwitch_1.out[1];
   DefaultRack_1.out[2] --> prueba_channel_3 --> DefaultSwitch_1.in[2];
   DefaultRack_1.in[2] <-- prueba_channel_3 <-- DefaultSwitch_1.out[2];
   DefaultRack_1.out[3] --> prueba_channel_3 --> DefaultSwitch_1.in[3];
   DefaultRack_1.in[3] <-- prueba_channel_3 <-- DefaultSwitch_1.out[3];
```

```
// -------------------------------------------------------------
   //    Connection between DefaultRack_0 and DefaultSwitch_1
   // -------------------------------------------------------------
   DefaultRack_0.out[0] --> prueba_channel_4 --> DefaultSwitch_1.in[4];
   DefaultRack_0.in[0] <-- prueba_channel_4 <-- DefaultSwitch_1.out[4];
   DefaultRack_0.out[1] --> prueba_channel_4 --> DefaultSwitch_1.in[5];
   DefaultRack_0.in[1] <-- prueba_channel_4 <-- DefaultSwitch_1.out[5];


   // -------------------------------------------------------------
   //    Connection between DefaultSwitch_0 and DefaultSwitch_1
   // -------------------------------------------------------------
   DefaultSwitch_0.out[6] --> prueba_channel_5 --> DefaultSwitch_1.in[6];
   DefaultSwitch_0.in[6] <-- prueba_channel_5 <-- DefaultSwitch_1.out[6];

endmodule
```

Ilustración 52 Conexiones establecidas recogidas en el fichero scenario.ned

ANEXO III BIBLIOGRAFÍA

SIMCAN: A Highly Configurable Simulation Framework for HPC Architectures and Applications

[Alberto Núñez, Javier Fernández, Jesús Carretero]

The XML DOM – Advanced

http://www.w3schools.com/xml/xml_dom_advanced.asp

XML Validation and XPath Evaluation

http://java.sun.com/developer/technicalArticles/xml/validationxpath/

Using Layout Managers

http://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

Controlling Rendering Quality

http://docs.oracle.com/javase/tutorial/2d/advanced/quality.html

Mouse drag and drop to draw

http://www.java2s.com/Code/Java/2D-Graphics-GUI/Mousedraganddroptodraw.htm